Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

GRADO EN INGENIERÍA INFORMÁTICA

Título del proyecto:

DESARROLLO DE UN MECANISMO DE COMBINACIÓN DE CLASIFICADORES BASADO EN LOS VECTORES DE SALIDAS MÁS CERCANOS

Alumno: Mikel Aingeru Jorge Soteras

Tutor: Mikel Galar Idoate

Pamplona, 25 de Junio de 2014

Desarrollo de un mecanismo de combinación de clasificadores basado en los vectores de salidas más cercanos

Mikel Aingeru Jorge Soteras

25 de Junio de 2014

Agradecimientos

Quiero agradecer la inmensa ayuda recibida por parte de mi tutor Mikel Galar Idoate, el tutor 10. Sin él este proyecto no hubiera tenido fin, y de tenerlo no habría sido el buen resultado que ha demostrado poder llegar a ser. Siempre dispuesto a atender la enorme cantidad de dudas que van surgiendo y siempre dispuesto a atenderte en el despacho sea la hora que sea. Muchísimas gracias tanto por la ayuda como por dejarme participar en este gran proyecto.

A mi hermano Javier Jorge Soteras, que ha tenido que soportar las inmensas “chapas” que le metía con mi proyecto y los resultados que iba obteniendo.

A mi novia Andrea por soportarme cuando estaba cardíaco porque no tenía tiempo o tenía que lograr de alguna forma que las ejecuciones fueran bien y por dejarla más de lado.

Por último y no menos importante a mis amigos Erik, Esparza, Sebas, Jose, Berjano, Ruben, Ibai etc. Por animarme las comidas en la universidad, la compañía durante los cafés, el sacarme de casa o de la biblioteca a tomar el aire y el ánimo cuando pensaba que era imposible que llegase a terminar este proyecto.

Índice general

1.- Introducción

Este proyecto consiste en desarrollar un nuevo modelo de combinación de clasificadores basado en la similitud entre las salidas que se obtienen entre el ejemplo a clasificar y los ejemplos de entrenamiento. Es decir, consiste en la aplicación del algoritmo de los k vecinos más cercanos para sustituir la fase de agregación clásica a la hora de decidir la clase a la que pertenece una instancia cuando se trabaja con conjuntos de clasificadores clásicos.

1.1 Sistemas de clasificación

Existen numerosos problemas que requieren clasificar cierta información representada por ejemplos o instancias, en clases o categorías. En el ámbito del aprendizaje automático, el objetivo de la clasificación consiste en aprender a discernir cual es la clase a la que pertenecen los ejemplos nuevos sin etiquetar. Para ello, se construye un clasificador que se entrena a partir del conocimiento sobre las instancias previas de las cuales se conoce la clase a la que pertenecen.

La aplicación de estos sistemas abarca todo tipo de ámbitos. Desde la identificación de caracteres en sistemas de reconocimiento óptico de caracteres (OCR), la decisión de concesión de créditos bancarios, la determinación de si un paciente tiene cierta enfermedad en medicina, hasta los controles de calidad en fabricación.

Clasificación supervisada y no supervisada

Dentro del aprendizaje automático existen dos tipos de clasificación: supervisada y no supervisada.

La clasificación supervisada cuenta con un conocimiento a priori del problema, es decir, cuenta con un conjunto de ejemplos de los cuales ya conocemos su clasificación. Está compuesta de dos fases: la primera, la fase de entrenamiento del clasificador y la segunda, la fase de clasificar una nueva instancia del problema.

La clasificación no supervisada, en cambio, no cuenta con un conocimiento a priori del problema por lo que entre otras tareas es necesario determinar el número de clases que queremos establecer.

1.2 Ensembles

La dificultad de los problemas de clasificación varía de unos a otros. Para suplir esta necesidad de mejora ante los casos en los que la clasificación es complicada de realizar correctamente con un único clasificador, surgieron los conjuntos de clasificadores o ensembles.

Esta técnica tiene por objetivo combinar el resultado de varios clasificadores para dar una respuesta final ante una nueva instancia de entrada. Esto es, realiza una primera fase de clasificación utilizando cada uno de los clasificadores que forman el ensemble para, en una segunda fase, agregar todos los resultados obtenidos por cada uno de los clasificadores.

1.3 Problema de clases no balanceadas

En sistemas de clasificación, se dice que un dataset está no balanceado cuando el número de instancias que representan una clase es menor que las que representan otras clases. Además, la clase que menor número de instancias tiene suele ser la clase que más interesa desde el punto de vista del aprendizaje automático. Este problema tiene mucho interés debido a que sucede en muchas situaciones reales en el conjunto de los problemas de clasificación, como en la detección de polución, control del riesgo, detección de fraudes y especialmente en el diagnóstico médico. En estos casos, que son problemas complejos, se ha podido comprobar que los ensembles permiten mejorar los resultados obtenidos por clasificadores individuales.

1.4 Algoritmo de los k vecinos más cercanos

A la hora de clasificar las instancias de un dataset, existen diversos métodos que podemos aplicar, entre los que se encuentran: k vecinos más cercanos, ID3, C4.5, lineal, máquinas de vectores soporte y perceptrón o perceptrón multicapa entre otros.

El algoritmo de los k vecinos más cercanos es un algoritmo de clasificación supervisada. Tal y como indica su nombre, trata de clasificar el ejemplo recibido como entrada a través de buscar en un conjunto de datos, el cual contiene las clases a las que pertenecen sus instancias, las k instancias más parecidas a la instancia de entrada para finalmente agregar las clases de las instancias similares con el fin de obtener una única clase como predicción.

1.5 Algoritmos genéticos

Los algoritmos genéticos son una técnica adaptativa de búsqueda y optimización. Están basados en la teoría de la evolución de Darwin, en la que una población a lo largo de generaciones evoluciona acorde a las leyes de la selección natural y la supervivencia de los más fuertes.

En el mundo real los individuos compiten entre sí en la búsqueda de recursos. Los que lo consiguen tienen una mayor probabilidad de generar mayor cantidad de descendencia, mientras que los que no lo consiguen tienen un menor número de descendientes o no los tienen. En la descendencia se transmiten los genes de los individuos mejor adaptados, que los nuevos individuos a su vez, también transmitirán a sus descendientes. De esta forma los genes con mejores características se propagan a un número cada vez mayor de individuos. La idea consiste en que la combinación de buenos genes de diferentes predecesores puede producir “súper individuos" con una mayor adaptabilidad. Así, el proceso evolutivo consigue que los nuevos individuos se adapten mejor a su entorno.

Los algoritmos genéticos tratan de imitar este comportamiento, donde la población de individuos son las posibles soluciones de un problema concreto. Cada uno de estos individuos es evaluado y obtiene una puntuación relacionada con la bondad de dicha solución (en el mundo real sería lo efectivo que es el individuo compitiendo por los recursos). Cuanto mejor sea la adaptación de un individuo, mayor serán sus posibilidades de reproducción con otro individuo que también ha sido seleccionado. Con el cruce de ambos se obtienen descendientes que comparten y combinan las características de sus padres. Si un individuo no se adapta, su probabilidad de reproducción es menor, y por tanto, la transmisión de sus genes a futuras generaciones. De esta forma se consigue que con cada generación se aumente la proporción de buenas características dentro de la población y se exploten las áreas más prometedoras del espacio de búsqueda, forzando así la convergencia hacia la solución óptima.

La potencia de los algoritmos genéticos viene del hecho de ser una técnica robusta capaz de abordar una amplia variedad de problemas. Aunque no se puede garantizar que se encuentre la solución óptima, existe una evidencia empírica de que se obtienen soluciones muy buenas. En general, el campo de aplicación de los algoritmos genéticos lo constituyen los problemas para los que no existe un método concreto especializado. En estos casos, los algoritmos genéticos funcionarían, pero lo más común es que los métodos especializados sean más eficientes ya que se han construido para dicha labor. Aplicaciones típicas de los

algoritmos genéticos son por ejemplo la optimización de la ruta en el problema del viajante, o la obtención de la combinatoria mínima para encontrar la solución en el juego MasterMind, pero también podemos encontrarnos con ellos en ámbitos más comerciales, como en problemas de diseño de horarios de aeropuertos, o el diseño de circuitos impresos para microchips.

1.6 Objetivos del proyecto

Como hemos mencionado, una forma de mejorar los sistemas de clasificación es a través de los ensembles. La fase de clasificación una vez obtenidos los clasificadores base se compone de dos fases:

1. La clasificación de la nueva instancia por parte de cada uno de los clasificadores base.
2. La agregación de los resultados obtenidos para asignar una clase final a la instancia.

En este proyecto, principalmente, nos centramos en la segunda fase, la de agregar los resultados obtenidos por los clasificadores que componen el ensemble para tratar de obtener la clase correcta de la instancia. Sin embargo, lo vamos a hacer de una forma diferente a la habitual que consiste en aplicar el método del voto o el voto ponderado (cada clasificador vota por una de las clases y la que más votos obtiene es la que se predice). Nuestra propuesta consiste en aplicar el algoritmo kNN para mejorar los resultados obtenidos. Esto en sí no tiene una gran complejidad y no mejoraría el modelo en gran medida. Por ello, aprovechando el uso del algoritmo kNN, nuestro objetivo también es realizar una reducción de clasificadores necesarios y por tanto mejorar el tiempo de respuesta del ensemble obtenido. Estos dos objetivos, necesariamente nos conducen a una reducción de instancias del conjunto de datos de train. Con esto, esperamos lograr mejorar los resultados obtenidos en la fase de agregación del ensemble original. Para terminar, trataremos con conjuntos de datos no balanceados, ya que es en estos casos en los que es más difícil clasificar, sobre todo los ejemplos de la clase minoritaria.

Para comprobar la efectividad de los resultados obtenidos en el proyecto, los compararemos con la estrategia de agregación más habitual, la del voto ponderado, mediante un amplio estudio estadístico que nos permita corroborar la validez de los resultados, o en caso contrario, desechar la metodología propuesta.

2.- Técnicas utilizadas

A continuación explicaremos en mayor detalle la estructura y funcionamiento de todas las técnicas aplicadas en el proyecto. En la sección 2.1 explicaremos el problema de las clases no balanceadas en el que nos centramos en este proyecto. La sección 2.2 presenta los ensembles junto con los diversos métodos de estos aplicados en este proyecto. La sección 2.3 explica el algoritmo kNN en el que nos basamos para mejorar la fase de agregación del ensemble. Para terminar, en la sección 2.4 explica los algoritmos genéticos y la aplicación de los mismos en este proyecto.

2.1 Problema de clases no balanceadas

Tal y como hemos explicado con anterioridad, el problema de clases no balanceadas aparece en los conjuntos de datos cuando la cantidad de instancias de las diferentes clases del problema no es similar, es decir, existe una mayor proporción de instancias de unas clases que de otras.

Como en nuestro caso trabajamos con clasificadores binarios, reducimos el problema al caso en el que existen muchas más instancias de una clase que de otra, lo que nos genera una clase mayoritaria, con un número mayor de representaciones, y una clase minoritaria, con una cantidad menor. Es importante destacar que generalmente la clase de interés acostumbra a ser la clase minoritaria.

Llevando al extremo el problema de las clases no balanceadas, es muy fácil ver el problema que generan. Pongamos un ejemplo: supongamos que en el aprendizaje automático de un clasificador contamos con un dataset compuesto con un 99% de instancias de la clase 1 y un 1% de instancias de la clase 0. Tomando como medida de efectividad del clasificador el porcentaje de aciertos, con un clasificador que clasifique todas las instancias como de la clase 1, independientemente de los atributos de las instancias, obtendríamos un porcentaje de acierto de un 99%. Aplicando esto a un posible caso real, si tenemos 10.000 de personas de las cuales 100 tienen cáncer, nuestro clasificador nos determinaría que ninguna de ellas tiene la enfermedad por lo que no trataríamos a las 100 personas afectadas. Si el objetivo de este clasificador es discernir si una persona tiene cáncer o no y siempre

predice que no, en este caso acertaría un 99% de las veces y fallaría solo un 1%. ¿Es este clasificador bueno? No, es pésimo ya que nunca nos ayudaría a predecir si una persona tiene cáncer o no.

Para mejorar la situación mostrada en el ejemplo anterior, podemos modificar la forma en que evaluamos la calidad de los resultados obtenidos. En vez de evaluar el total de aciertos, lo que resulta en una información sesgada en el caso de los conjuntos de datos no balanceados, hay que evaluar el total de aciertos o fallos sobre cada clase [1]. Así, lograríamos un modelo con cuatro factores, en el caso de un clasificador binario, como el mostrado en la tabla 1.

Tabla 1

	Positive prediction	Negative prediction
Positive class	True Positive (TP)	False Negative (FN)
Negative class	False Positive (FP)	True Negative (TN)

- True positive TP: Instancias de la clase positiva bien clasificadas.
- True negative TN: Instancias de la clase negativa bien clasificadas.
- False positive FP: Instancias de la clase negativa mal clasificadas.
- False negative FN: Instancias de la clase positiva mal clasificadas.

Con estos factores, podríamos obtener nuevas funciones de evaluación a través de generar nuevas medidas como la media geométrica o Gmean (1), la Fmean (2) o el AUC (*Area Under Curve, 3*):

(1) $Gmean = \sqrt{TPrate * TNrate}$, donde TPrate y TNrate son:

$$TPrate = \frac{TP}{TP + FN} \qquad TNrate = \frac{TN}{TN + FP}$$

(2) $Fmean = \frac{2*recall*precision}{recall+precision}$, donde recall y precision son:

$$recall = \frac{TP}{TP+FN} \qquad precision = \frac{TP}{TP+FP}$$

(3) $$AUC = \frac{TPrate+TNrate}{2}$$

Para comprobar la diferencia entre estas funciones de evaluación y la anterior, tomemos como ejemplo la media geométrica. Con esta función de evaluación, en el ejemplo anterior el resultado obtenido de clasificar todo como 1 sería un 0. Como se puede observar es una medida muy diferente de la obtenida antes y demuestra la poca efectividad del clasificador obtenido.

Con estas nuevas funciones de evaluación podremos obtener una medida distinta y a la vez más útil para el problema de los conjuntos de datos de clases no balanceadas.

2.2 Ensembles

Como ya hemos explicado en la sección 1.2, los ensembles son conjuntos de clasificadores [2]. Se pueden aplicar cualquier tipo de clasificadores a la hora de crear el conjunto, ya sean supervisados o no supervisados. Para nuestro caso, utilizaremos un ensemble de clasificadores supervisados ya que nos centramos en el problema de clasificación.

La estructura de un ensemble es muy sencilla. Podríamos imaginárnosla como un vector en el cual cada posición contiene un clasificador.

Por otra parte, su funcionamiento puede variar dependiendo de la técnica elegida. Por lo general, todas se basan en aplicar un mismo algoritmo de clasificación repetidas veces con ligeras diferencias en cada una de ellas. Además, también puede variar la forma de agregar las salidas para obtener una clase final.

Para entender mejor el concepto y las partes de un ensemble, vamos a proceder a explicar el clasificador tomado como base durante el desarrollo de este proyecto, las técnicas empleadas en el ensemble y finalmente como hemos desarrollado la fase de agregación.

2.2.1 Algoritmo C4.5

C4.5 es un algoritmo de aprendizaje supervisado utilizado en los problemas de clasificación en aprendizaje automático y minería de datos [3]. Se trata de un algoritmo que genera un árbol de decisión utilizando el conjunto de datos de train, para a través de él clasificar las nuevas instancias recibidas. En el árbol generado, existen tres tipos de elementos distintos:

1. Nodo: Es un elemento del árbol en el cual elegiremos un atributo sobre el que evaluar el ejemplo a clasificar. Del nodo salen varias ramas cada una teniendo en cuenta diferentes valores del atributo.
2. Rama: Es un elemento del árbol en el que se elegirá con qué valor o valores del atributo se accederá al siguiente nodo u hoja.

3. Hoja: Es un elemento final del árbol. De ella no parten más ramas y es la encargada de predecir una clase para los ejemplos que hayan recorrido el camino a través del árbol hasta ella.

Descripción

Tal y como hemos mencionado, el C4.5 genera un árbol de decisión. Esto es, genera una estructura de tipo árbol. Para ello se utiliza como algoritmo el mostrado en la figura Algoritmo 1.

Algoritmo 1

INPUT: an attribute-valued dataset D
1: Tree = { }
2: **IF** D is "pure" **OR** other stopping criteria met **THEN**
3: terminate
4: **END IF**
5: **FOR** all atribute a ∈ D **DO**
6: Compute information-theoretic criteria if we Split on a
7: **END FOR**
8: a_{best} = Best attribute according to above computed criteria
9: Tree = Create a decision node that tests a_{best} in the root
10: D_v = Induced sub-datasets from D based on a_{best}
11: **FOR** all D_v **DO**
12: $Tree_v$ = C4.5(D_v)
13: Attach $Tree_v$ to the corresponding branch of Tree
14: **END FOR**
15: **RETURN** Tree

El algoritmo se basa en generar una estructura de tipo árbol. Para lograrlo, hace una serie de llamadas recursivas. En cada una de estas llamadas, lo que hace es calcular el ratio de ganancia obtenida por cada atributo del dataset (4), utilizando como medida de la ganancia (5) y de la entropía (6).

(4) $Gain(Attrib) = Entropy(class = 1\ in\ D) - \sum_v \frac{|D_v|}{|D|} Entropy(class = 1\ in\ D)$

Donde v es el conjunto de posibles valores del atributo, D es el conjunto de datos completo, D_v es el subconjunto del conjunto de datos para cada valor que tenga ese atributo v, y la notación |·| denota el tamaño del dataset (número de instancias).

(5) $GainRatio(a) = \frac{Gain(a)}{Entropy(a)}$

(6) $Entropy(a) = \sum_{i=1}^{c} -p_i\, log_2 p_i$ Donde p_i son las probabilidades de que tome un atributo el valor i.

Una vez calculadas las ganancias de los diferentes atributos, se procede a elegir el atributo que mayor ganancia aporte. Teniendo este como base, se genera un nodo del árbol con ese atributo. Después, se divide el data-set en dos (o más si el atributo es nominal) conjuntos de datos nuevos que son los que se envían por cada rama del árbol.

Después, se procede a realizar las llamadas recursivas al propio método para generar los siguientes sub-niveles del árbol, una por cada dataset generado. Al mismo tiempo, habiendo recibido el sub-árbol generado por la llamada recursiva, lo incluye en el árbol creado, en el cual la primera vez que hacemos una llamada recursiva solo existe el nodo raíz que está compuesto por el atributo seleccionado que proporciona mayor ganancia.

Para terminar, devuelve el árbol generado a través de toda la serie de llamadas recursivas.

Características

Este algoritmo puede trabajar con cualquier tipo de atributos, ya sean numéricos, booleanos o categóricos. Además, dependiendo del tipo, se pueden plantear formas distintas de abordar este problema. Para el caso de los numéricos, podemos dividir el conjunto de datos en tantos conjuntos como queramos (normalmente 2). En el caso de los booleanos, se separan dos conjuntos como es lógico, los verdaderos y los falsos. Y para el caso de los atributos categóricos, creamos tantos conjuntos como valores posibles.

Para el caso de como seleccionar el atributo por el que dividir el árbol, se utiliza el ratio de ganancia. Aun así, se podría aplicar la ganancia pero esto generaría una tendencia de favorecer a las pruebas con muchos resultados.

A la hora de terminar la ejecución recursiva del algoritmo, se pueden tener en cuenta varias opciones. Una de ellas es que todas las instancias del dataset sean puras, esto es, que pertenezcan a una misma clase por lo que la hoja tomaría el valor de la clase de las instancias. Otra forma, es tener en cuenta el número de instancias a clasificar. Si se trata de un número muy reducido, se terminara la generación de sub-árboles por debajo del nodo y se asignara a la hoja resultante el valor de la clase mayoritaria del conjunto de instancias. Además, ambas técnicas se pueden aplicar al mismo tiempo, comprobando primero si las instancias del dataset son puras y segundo si no superan un umbral establecido. Para terminar, falta decir que precisamente este grado de pureza es el utilizado como salida del clasificador para usar en el ensemble y lo calculamos con el modificador de Laplace. Lo modificamos de esta forma

para no valorar de igual forma las hojas que, aunque puras, tengan pocos elementos ante las que tengan muchos.

Poda de árboles

La poda de árboles es un método aplicado después de haber generado el árbol. Sirve para evitar el sobre-aprendizaje del árbol. Existen diversos métodos de realizar la poda pero en el ensemble en el que nos basamos en este proyecto solo se aplica uno de ellos. El método aplicado consiste en estimar el error obtenido por la rama más grande, estimar el error obtenido por todo el sub-árbol y estimar el error si el nodo se convirtiese en hoja. Una vez calculadas las estimaciones, dependiendo de los resultados obtenidos se actúa de diferentes formas. Para el caso en el que el error estimado de convertir el nodo en hoja sea el menor de los tres, se sustituye el nodo por una hoja. Si el caso anterior no se da, si el error estimado de la rama más grande es menor que el error estimado por el sub-árbol entero, se sustituye el nodo actual por el nodo siguiente de la rama más grande. Si no se da ninguna de estas situaciones, no se aplica cambio alguno, por lo que no se aplica ninguna poda.

Una de las cosas más importantes de este método es que, a diferencia de otros métodos posibles, no requiere un conjunto de datos de test a parte para realizar la poda ya que se basa en estimaciones de errores.

2.2.2 Técnicas del ensemble

En esta sección vamos a explicar las diferentes técnicas tenidas en cuenta en este proyecto, las cuales son efectivas para el caso de los datasets no balanceados [1]. Todas ellas están basadas en dos técnicas de ensembles:

1. **Bagging:** Esta técnica está basada en generar diferentes conjuntos de datos, tomando como datos subconjuntos de datos del conjunto total de train, sobre los que entrenar cada uno de los clasificadores de forma independiente.
2. **Boosting:** Esta técnica se basa en dar más importancia a las instancias mal clasificadas por los clasificadores anteriores a la hora de crear otro clasificador para el ensemble.

Además, para el caso de los conjuntos de datos no balanceados, es necesario combinar estas técnicas con técnicas de pre-procesamiento de datos. De esa forma surgen los ensembles para conjuntos de datos no balanceados que son los siguientes.

RUSBoost

Es un método de boosting que aplica un pre-procesamiento al dataset de entrada al clasificador. Combina la técnica de boosting con la técnica de pre-procesamiento de datos de undersamplig (bajo-muestreo aleatorio). Modifica el dataset eliminando instancias de la clase mayoritaria de forma aleatoria. Además, normaliza los pesos de las instancias restantes en el dataset con respecto al total. Después de entrenar un clasificador, los pesos de las instancias del dataset original son actualizados.

SMOTEBagging

Se trata de una combinación entre un método de bagging que genera diferentes conjuntos de instancias a partir del dataset inicial, y el método SMOTE de pre-procesamiento de datos. Para el entrenamiento de cada clasificador utiliza un conjunto distinto. A la hora de generar estos conjuntos, introduce la misma cantidad de ejemplos de cada clase pero se establece una tasa de remuestreo distinta en cada iteración, y esta tasa es la que define el número de instancias positivas reemplazadas aleatoriamente, con reemplazamiento, del dataset original en cada iteración, mientras que el resto se generan con el algoritmo SMOTE.

UnderBagging

Como en el caso de SMOTEBagging, es la combinación de bagging, por lo que generará diferentes conjuntos de instancias para entrenar los diferentes clasificadores, y de undersamplig para el pre-procesamiento de los datos. En este caso, a la hora de crear el nuevo conjunto de instancias lo que hace es eliminar instancias aleatorias de la clase mayoritaria y mantiene todas las instancias de la clase minoritaria.

EasyEnsemble

Este caso es el más particular de todos. Es un método basado en bagging pero que para entrenar los clasificadores con los conjuntos generados utiliza AdaBoost, el cual es un método de boosting. Para la primera parte, el bagging, utiliza el método de UnderBagging, mientras que para la segunda, utiliza el AdaBoost sin aplicar ninguna operación a las instancias del conjunto después de cada iteración.

2.3 Algoritmo kNN

Tal y como indicábamos en la introducción, este algoritmo es un algoritmo de aprendizaje supervisado que se basa en el cálculo de distancias entre el ejemplo a clasificar y un conjunto de referencia

(generalmente el conjunto de entrenamiento) [4]. Un punto a tener en cuenta de este algoritmo es que en vez de realizar una fase de aprendizaje propiamente dicha sobre las instancias del conjunto de aprendizaje, se basa en el conjunto de aprendizaje para generar la clase predicha de cada instancia que obtiene como entrada.

Descripción

El algoritmo kNN siempre guarda el conjunto de instancias de entrenamiento completo para realizar la clasificación. Para realizar la clasificación, sigue el Algoritmo 2.

Algoritmo 2

INPUT: D, the set of training objects, the test object **z**, which is a vector of attribute values, and L, the set of clases used to label the objects
OUTPUT: $c_z \in L$, the class of z
1: **FOREACH** object **y** ∈ D **DO**
2: Compute d(z,y), the distance between **z** and **y**
3: **END**
4: Select N ∈ D, the set (neighborhood) of k closest training objects for z
5: $c_z = argmax_{v \in L} \sum_{y \in N} I(v = class(c_y))$ where I(·) is an indicator function that returns the value 1 if its argument is true and 0 otherwise

Como podemos ver, el algoritmo es muy sencillo. Recibe como entrada un conjunto de referencia, generalmente de entrenamiento, una instancia a clasificar y un vector que contiene las diferentes clases posibles del conjunto.

Para cada instancia del conjunto de train, calculamos su distancia con la instancia nueva a clasificar. Después seleccionamos las k instancias más cercanas a la instancia a clasificar.

Para terminar, agregamos los resultados obtenidos en las distancias de las k instancias seleccionadas para obtener una predicción de la clase final.

Cuestiones

Una de las cuestiones a tener en cuenta en este algoritmo es la elección de k. Dependiendo del valor de k que se tome la efectividad del algoritmo puede variar en mayor o menor medida. Tal y como se muestra en la figura 1 el valor de la k determinará si la clasificación es correcta o no. Además, no hay manera de saber para qué valor de k obtendremos una mayor precisión en los resultados.

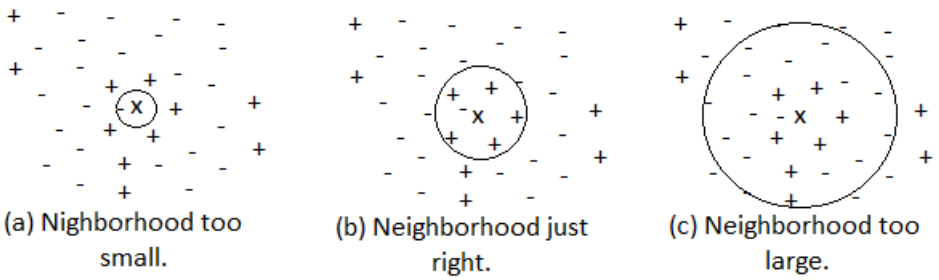


Figura 1 clasificación kNN con pequeño, mediano y gran valor de k

Otra de las cuestiones a tener en cuenta es como realizar la agregación final del algoritmo. Esta agregación puede llevarse a cabo a través del voto, el cual predice la clase con mayor número de instancias entre las k seleccionadas, o del voto ponderado, el cual pondera la información aportada por cada instancia de las k seleccionadas con la distancia entre dichas instancias y la nueva instancia a clasificar. El aplicar el voto ponderado nos supondría otra cuestión como la de elegir como ponderar la información aportada por cada instancia con su distancia. En el caso de este proyecto aplicaremos el voto, sin ponderar.

La elección de la medida de distancia es otra cuestión a tener en cuenta. Normalmente se aplica la distancia Euclidea (7) o Manhattan (8). Aun así, estas y otras distancias pueden ser aplicadas para calcular la distancia entre dos puntos [8]. En nuestro caso como tratamos con atributos numéricos, usamos la distancia Euclidea por ser la más comúnmente utilizada.

$$(7)\ d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

$$(8)\ d(x,y) = \sqrt{\sum_{k=1}^{n}|x_k - y_k|}$$

2.4 Algoritmos genéticos

A continuación explicamos en mayor detalle la estructura y funcionamiento de los algoritmos genéticos, así como el papel que tienen en este proyecto.

2.4.1 Introducción a los algoritmos genéticos

Como ya hemos explicado en la Sección 1.1.5, los algoritmos genéticos imitan la evolución natural. Una población inicial evoluciona y se adapta para obtener los mejores resultados. Esta evolución se lleva a cabo cruzando los individuos de la población, y en algunos casos, por mutaciones. Los individuos (o cromosomas) se representan como

cadenas de valores (o genes), donde cada uno hace referencia a un parámetro que se desea ajustar.

Un aspecto importante a tener en cuenta es la representación de la información, la forma en la que se definen los parámetros a ajustar dentro del cromosoma. El conjunto de valores que forma un cromosoma, es decir, la representación interna de la solución se denomina genotipo. La solución que forma el cromosoma es el fenotipo. Un ejemplo de genotipo y fenotipo se muestra en la figura 2, donde se ve una posible representación del problema del viajante.

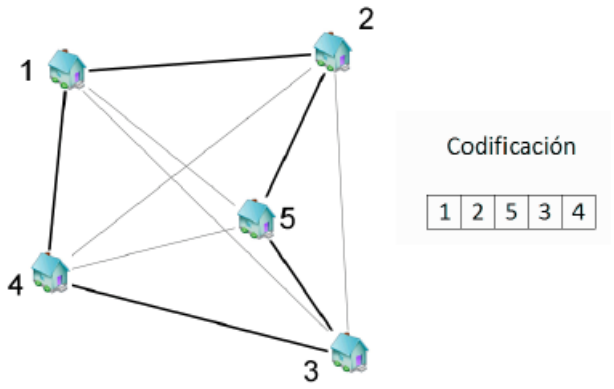


Figura 2 Fenotipo y Genotipo del problema del viajante

Un algoritmo genético básico tiene las siguientes etapas:

- **Inicialización:** se genera una población inicial de cromosomas de forma aleatoria.
- **Evaluación:** se evalúa cada cromosoma aplicándole una función objetivo o *fitness*, que indica como de buena es la solución respecto al problema que queremos resolver.
- **Selección:** se seleccionan los cromosomas para ser cruzados dos a dos. Existen muchos criterios de selección, algunos de ellos son:
    - Selección elitista: se garantiza la selección de los miembros más aptos de cada generación.
    - Selección proporcional a la aptitud: los individuos más aptos tienen más probabilidad de ser seleccionados, pero no la certeza.
    - Selección por rueda de ruleta: la probabilidad de ser seleccionado un individuo es proporcional a la diferencia entre su aptitud y la de sus competidores.
- **Cruce:** consiste en que dos cromosomas padre combinen sus genes para crear un nuevo individuo que comparta las características de ambos. La elección del operador de cruce es

importante en el resultado de la evolución. Algunos operadores de cruce comunes son:

- Cruce básico: se elige un punto aleatorio de la cadena. La parte anterior del punto es copiada del cromosoma de un padre y la posterior del otro. En la figura 3 vemos un ejemplo del cruce básico.
- Cruce multipunto: el procedimiento es igual que en el cruce básico, a diferencia de que existe más de un punto de cruce. En la figura 4 podemos ver un ejemplo de cruce multipunto.
- Cruce uniforme: para cada gen de la cadena del descendiente existe una probabilidad de que el gen pertenezca a uno de los padres, y otra probabilidad de que se obtenga del otro padre. En la figura 5 vemos un ejemplo del cruce uniforme con una probabilidad del 75%.

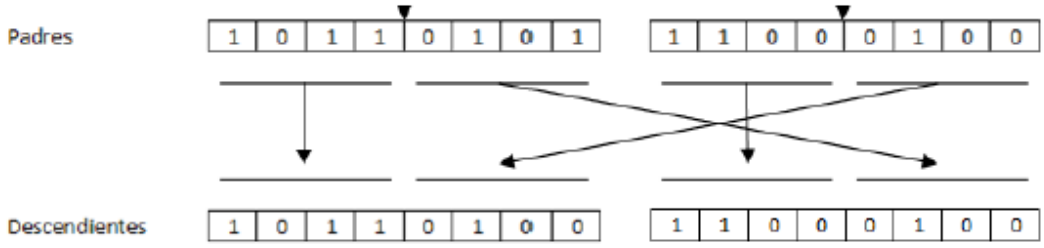


Figura 3 Operador de cruce básico

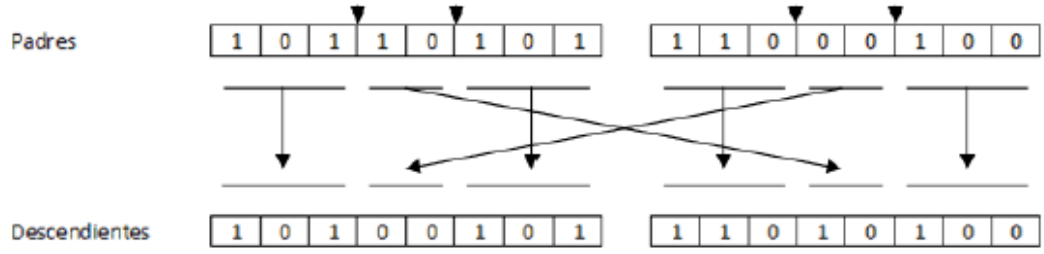


Figura 4 Operador de cruce multipunto

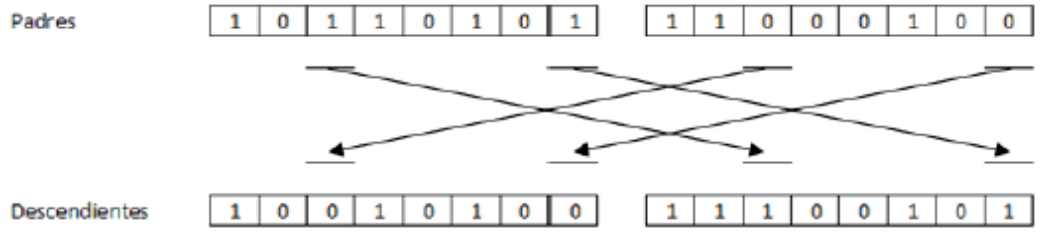


Figura 5 Operador de cruce uniforme con probabilidad 75%

- **Mutación:** no siempre se realiza, existe una probabilidad de que esto ocurra, y consiste en la variación de forma aleatoria de al menos uno de los genes del individuo. Como resultado se obtienen nuevos individuos. En la figura 6 vemos un ejemplo de mutación.

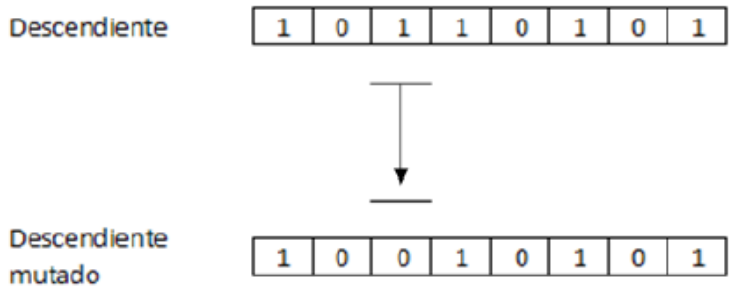


Figura 6 Ejemplo de mutación

- **Reemplazo:** una vez realizado el cruce y la mutación, los hijos más prometedores entran en la población. Podemos aplicar distintas estrategias de reemplazo, por ejemplo:
  - Reemplazar el Peor (*Replace Worst Strategy*): si un hijo mejora al peor individuo de la población actual, lo reemplaza. Es un criterio elitista con una alta precisión selectiva, incluso cuando sus padres son elegidos aleatoriamente.
  - Selección de Torneo Restringido (*Restricted Tournament Selection*): se buscan los individuos más parecidos a los nuevos descendientes. Si los descendientes son mejores, sustituyen al individuo en la población actual.
  - Reemplazar el Peor Entre Semejantes (*Worst Among Most Similar Replacement*): se compone de dos pasos. En el primero, se crean *f* grupos eligiendo aleatoriamente *g* individuos de la población. En el segundo, se selecciona para cada grupo aquel con mayor similaridad con el nuevo individuo. Se producen *f* individuos candidatos para el reemplazo, donde se selecciona aquel con peor valor, para ser reemplazado por el nuevo.
  - Algoritmo de Multitud Determinístico (*Deterministic Crowding*): se empareja cada descendiente con un padre de forma que las distancias entre padre-hijo sean menores. Si la solución del descendiente es mejor que la del padre asociado, el descendiente entra en la población sustituyendo a su padre.

La figura 7 esquematiza las etapas de un algoritmo genético. Estos pasos se repiten hasta que se satisface una condición de parada, que

normalmente es un número máximo de iteraciones o un número de iteraciones consecutivas en las que no hay cambios en la población. Con cada iteración se trata de conseguir que la población converja a una solución mejor. En el Algoritmo 3 se muestra el pseudo-código de un algoritmo genético

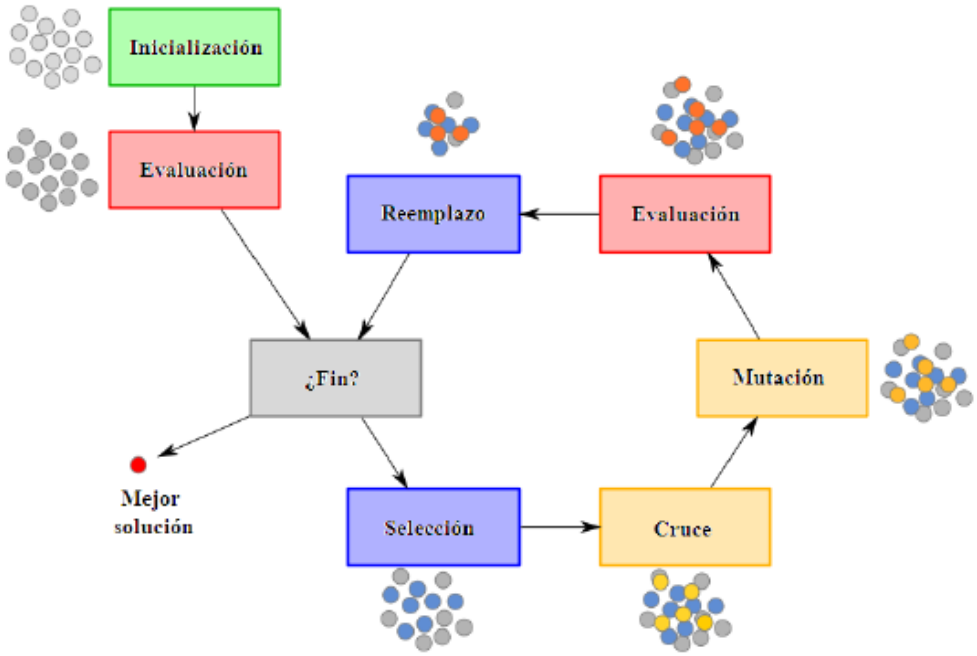


Figura 7 Etapas de los algoritmos genéticos

Algoritmo 3

```
Inicializar(Poblacion(0))
generacion = 0
Evaluar(Poblacion (0))
mientras (no CriterioParada) hacer
        Padres = Seleccion(Poblacion (generacion))
        Hijos = Operadores de Cruce(Padres)
        Operadores de Mutacion(Hijos)
        NuevaPob = Reemplazar(Hijos,Poblacion(generacion))
        generacion ++
        Poblacion(generacion) = NuevaPob
retornar Mejor Solucion Hallada
```

2.4.2 Algoritmo CHC

Este algoritmo genético destaca por su sencillez, a pesar de no ser un algoritmo complejo, con él conseguimos obtener muy buenos resultados. Es por esto que es utilizado en multitud de aplicaciones. Originalmente se basaba en una codificación binaria para la representación de los cromosomas pero actualmente también ha sido

adaptado a la codificación real. A continuación detallamos la versión binaria del algoritmo ya que es la aplicada a este proyecto.

CHC binario

El algoritmo CHC es una variante del algoritmo genético clásico que propuso Eshelman [5]. Fue uno de los primeros en mantener un buen equilibrio entre diversidad y convergencia. Se caracteriza por utilizar una estrategia de selección elitista y un operador de cruce uniforme (HUX). La selección elitista consiste en seleccionar los N mejores cromosomas entre padres e hijos y formar con ellos la siguiente generación de población. El operador de cruce uniforme (HUX) intercambia exactamente la mitad de los bits que son distintos entre los padres. Para prevenir el incesto (que cromosomas hermanos, y por tanto con genes parecidos, se crucen), este cruce solo se permite entre individuos cuya diferencia bit a bit (su distancia de Hamming) sea superior a un umbral de cruce, inicialmente un cuarto de la longitud del cromosoma. Este umbral disminuye en una unidad con cada generación que no crea ningún descendiente. En CHC no existe la mutación, la diversidad se consigue a través de un mecanismo de reinicio de la población que se aplica cuando el umbral de cruce es inferior a cero. Esto consiste en sustituir cada individuo de la población, a excepción del mejor, por uno aleatorio o una variación del mejor. Aquí exponemos el pseudo-código del algoritmo.

Algoritmo 4

```
Input: population
        Initialization(population)
        d = L / 4
        Evaluate(population)
        while termination condition not satisfied do
                candidates = SelectParents(population)
                offSpring = CrossParents(candidates)
                Evaluate(offSpring)
                SelectNewPopulation(population, offspring)
                If Population not changed then
                        d = d – 1
                end
                if d < 0 then
                        Restart(population)
                        Initialize(d)
                end
        end
Output: Best(population)
```

2.4.3 Reducción de instancias y selección de clasificadores

Debido a la gran cantidad de datos generados en muchos campos de investigación, el uso de algoritmos de aprendizaje automático se ha vuelto un reto. El uso de técnicas de reducción de información en las primeras fases de la construcción de clasificadores es una necesidad muy apremiante. El objetivo principal de estas técnicas es incrementar la eficiencia del proceso de clasificación, eliminando instancias redundantes, y reducir la tasa de error, eliminando instancias y características que generen ruido.

Tal y como hemos comentado, el algoritmo kNN no tiene una fase de entrenamiento per se, sino que utiliza todo el conjunto de train para clasificar una nueva instancia. Por tanto, la efectividad del algoritmo, aparte de depender de los parámetros del mismo, dependerá en gran medida de las instancias contenidas en el conjunto de train. Debemos destacar la ineficiencia del método para el caso en el que el conjunto de datos de train sea extremadamente grande. Para evitar este problema, además de para mejorar la precisión del clasificador generado con el nuevo conjunto de datos, se realiza una reducción de instancias y características a través de un algoritmo genético [6].

Tal y como hemos visto en los algoritmos genéticos, podemos definir nuestro cromosoma como un vector de ceros y unos teniendo en cuenta que cada gen con un valor a uno significa que la instancia o característica representada con ese gen se incluirá en el dataset final, mientras que un valor asignado a cero será descartado. Siendo así, podemos aplicar un algoritmo genético que se encargue de escoger la mejor combinación de instancias y características para nuestro dataset, teniendo como función de evaluación la tasa de aciertos con el dataset generado a partir de cada cromosoma con el algoritmo kNN.

Anteriormente hemos mencionado que se puede hacer tanto una reducción de instancias como reducción de características. A la hora de llevar esto a cabo contemplamos diversas formas de abordar este problema (Las últimas dos fueron contempladas durante el desarrollo del proyecto [7]):

1. **IS (Instance Selection):** Este método reduce únicamente las instancias del conjunto de entrenamiento. Con esto trata de eliminar instancias redundantes e instancias que aporten ruido al problema.

2. **FS (Feature Selection):** Este método reduce el número de características del conjunto. En el caso de este proyecto lo nombraremos CS (Clasifier Selection) ya que en nuestro caso la reducción de características es una reducción de clasificadores del ensemble. Este método tiene por objetivo reducir los atributos, o clasificadores en nuestro caso, que aporten poca información a la hora de clasificar.
3. **IS-FS (IS-CS para nosotros):** Este método es una combinación de los dos vistos anteriormente. Por tanto, lleva a cabo tanto una reducción de instancias como de características.
4. **MS (Majority Selection):** Este método reduce el número de instancias de la clase con mayor número de apariciones en el conjunto de entrenamiento. Con esto se pretende favorecer la aparición de instancias de la clase minoritaria, la que menos representación tenga en el conjunto. El objetivo de este método es el de mejorar el número de aciertos a la hora de clasificar la clase minoritaria.
5. **MS-FS (MS-CS para nosotros):** Este método es una combinación de los métodos MS y CS/FS vistos anteriormente por lo que, además de llevar a cabo una reducción del número de instancias de la clase mayoritaria, lleva a cabo una reducción del número de características que representan el conjunto de datos.

3.- Desarrollo y modelos planteados

En este capítulo, vamos a tratar de explicar el proceso llevado a cabo a la hora de desarrollar el proyecto. Para eso, vamos a dividir el proceso en siete fases principales. La sección 3.1 presenta la propuesta inicial a desarrollar en el proyecto. La sección 3.2 explica la fase en la que se modifica el ensemble sobre el que vamos a trabajar. La sección 3.3 presenta la fase de agregar utilizando el algoritmo kNN. La sección 3.4 es la fase de programación del algoritmo CHC. La sección 3.5 explica la optimización del algoritmo kNN. La sección 3.6 expone el resto de optimizaciones realizadas. La sección 3.7 es una fase de aclaración del código y realizar unos últimos añadidos. Por último, la sección 3.8 es la fase de preparación y ejecución de los lanzamientos y la agregación de los resultados obtenidos.

3.1 Propuesta inicial

El objetivo de este proyecto es modificar la fase de agregación del ensemble. En vez de recurrir al voto o al voto ponderado como es habitual, vamos a recurrir al algoritmo kNN para agregar las salidas producidas por el ensemble y establecer la clase final.

Para ello, vamos a transformar cada instancia inicial, tanto del conjunto de datos de test como de train, en otra distinta que viene dada por las confianzas obtenidas en los clasificadores que componen el ensemble. Con estas nuevas instancias ya calculadas, vamos a proceder a agregar los resultados utilizando el algoritmo kNN.

Para tratar de mejorar el modelo, ya que de la forma planteada hasta el momento no mejora los resultados obtenidos con el voto o el voto ponderado, vamos a llevar a cabo una reducción de instancias y de características. Para ello, vamos a utilizar un algoritmo genético, el CHC más concretamente, sobre las instancias trasformadas obtenidas como salida del ensemble. Con esta reducción, por una parte vamos a reducir las instancias que componen el conjunto de entrenamiento del algoritmo kNN mejorando la agregación final obtenida a través del mismo. Por otra parte, vamos a reducir el número de clasificadores necesarios en el ensemble suprimiendo los resultados obtenidos en dichos clasificadores del conjunto de datos de entrenamiento del algoritmo kNN.

3.2 Fase de modificación del ensemble

Para empezar, tomamos como partida el código de un ensemble configurable. Esto es, el código en Java de un ensemble basado en el algoritmo C4.5 capaz de trabajar con todos los métodos del ensemble explicados con anterioridad en esta memoria. Por lo tanto, ya partíamos de la base de poder ejecutar el ensemble con los métodos RUSBoost, SMOTEBagging, UnderBagging y EasyEnsemble con una cantidad configurable de clasificadores. Partiendo de este código, nuestra labor fue transformar las salidas en un nuevo dataset para poder trabajar sobre este más adelante.

Para lograrlo, además de generar los ficheros que ya se generaban, los cuales consistían en series de clases predichas y clases reales tanto en train como en test, tuvimos que generar nuevos ficheros para poder trabajar con ellos en fases posteriores del proyecto. Estos nuevos ficheros inicialmente eran cuatro:

1. “output.tra” y “output.tst”: Son los ficheros que contienen las confianzas de cada clasificador C4.5 del ensemble sobre cada instancia. Estas confianzas son modificadas para que en la clase mayoritaria se guarde la confianza del clasificador y en la minoritaria se guarde 1 menos dicha confianza. Esto nos facilita el trabajo en fases posteriores del proyecto. Estos ficheros son generados con la estructura necesaria para ser válidos para trabajar con la aplicación KEEL (*Knowledge Extraction based on Evolutionary Learning*). En la siguiente figura podemos ver la estructura de estos ficheros en un pequeño ejemplo.

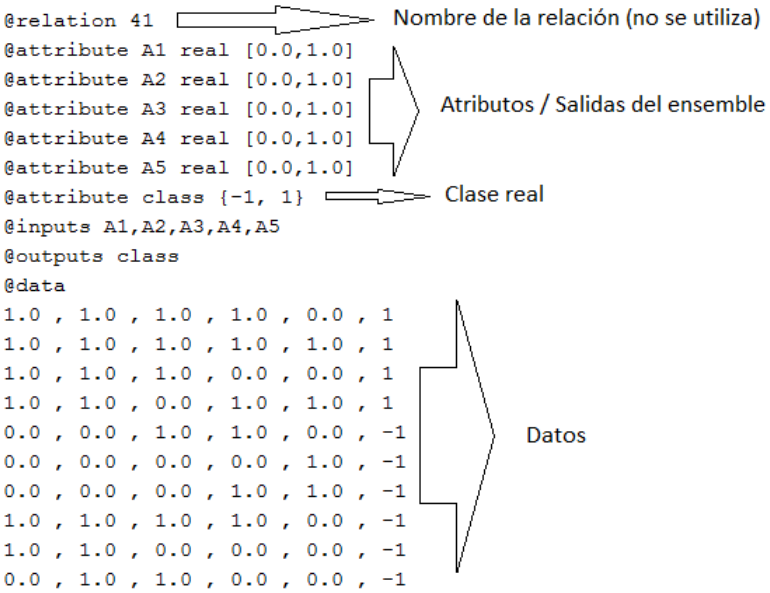


Figura 8 Ejemplo "output.tra" con estructura de KEEL

2. “alfaTrain.dat” y “alfaTst.dat”: Son ficheros que contienen los valores de los alfa, utilizados en las técnicas de boosting, tanto de train como de test.

Debido a que la siguiente fase del proyecto está codificada con Matlab, los nuevos ficheros generados “output.tra” y “output.tst” eran difíciles de leer. Por ello, al final se agregaron dos ficheros más a la cantidad de ficheros generados (“outputTra.dat” y “outputTst.dat”). Estos nuevos ficheros serían similares a los ficheros anteriores solo que no estarían preparados para trabajar con la aplicación KEEL ya que no seguirían su estructura, por lo que la lectura en Matlab sería muy sencilla y rápida.

3.3 Fase de agregación con kNN

En esta fase partimos desde cero. Lo que hicimos fue crear una primera versión funcional que fuera capaz de varias cosas.

En primer lugar tenía que ser capaz de leer los ficheros generados por el ensemble, tal y como hemos comentado hubo que generar dos ficheros más en el ensemble, y poder extraer los datasets de train y test con sus respectivas clases y los valores de alfa utilizados en las técnicas de boosting.

En segundo lugar, tenía que ser capaz de obtener los resultados de la clasificación obtenida con el ensemble a través de los datos obtenidos con los ficheros. Para ello tuvimos que desarrollar en Matlab la fase del ensemble que, una vez teniendo las confianzas de los clasificadores, clasificaba las instancias.

En tercer lugar, tenía que ser capaz de calcular la calidad de los resultados obtenidos utilizando las medidas explicadas en el apartado 2.1. En esta fase los resultados obtenidos se mostraban por pantalla para una mayor facilidad a la hora de comprobar los resultados obtenidos y el correcto funcionamiento del programa. Más adelante se transformarían estas salidas en salidas a ficheros para poder automatizar las ejecuciones del proyecto.

Por último y más importante, tenía que ser capaz de aplicar el algoritmo kNN sobre los conjuntos de train y test leídos. Para ello, tuvimos que programar el algoritmo, cosa que con las funciones específicas de Matlab para trabajar con matrices, como las de suma y el trabajo con índices, no fue un trabajo demasiado difícil. Además, el valor k aplicado en el algoritmo tenía que ser un parámetro del mismo.

3.4 Fase de programación del CHC

En esta fase nos centramos en lograr una versión que fuera capaz de aplicar el algoritmo CHC al dataset de train leído a partir del fichero. Para ello, hicimos una primera versión del algoritmo, no muy organizada ni muy clara pero sí funcional.

En esta fase de codificación del algoritmo CHC, hubo que codificar todas las funciones necesarias para ejecutarlo:

- **Evaluación:** Es la función encargada de, pasando la población del CHC, calcular el fitness de cada uno de los cromosomas. Esta tarea, aunque por apariencia sencilla, tiene cierta complejidad ya que tiene que ser capaz de recibir como parámetro el tipo de reducción a aplicar: *Instance* Selection (IS), *Feature* o *Clasifier Selection* (FS/CS), *Instance and Clasifier Selection* (IS-CS). Por tanto, dependiendo del tipo de reducción a aplicar, la evaluación varía. Como medida del fitness, se utilizaron, de forma configurable a través de un parámetro "tipoFitness", las medidas de Gmean, Fmean y AUC comentadas en el apartado 2.1.
- **Cruce:** Esta función es la encargada de realizar el cruce HUX explicado anteriormente. Al igual que la función anterior, necesita de un parámetro *prob0to1* para indicar la probabilidad de que un bit durante el cruce pase de ser 0 a ser 1.
- **Reinicio:** Esta función es la encargada de llevar a cabo el reinicio de la población manteniendo el 35% del mejor individuo de la población en cada nuevo individuo, además de mantener al mejor individuo al mismo tiempo.

Por otra parte, hubo que codificar el algoritmo completo utilizando estas funciones además de muchas más ya implementadas en Matlab. Otros parámetros añadidos en la función del CHC fueron los siguientes: cantidad de la población a tratar en el algoritmo genético (L), número de iteraciones máximas a realizar (num_iteraciones) o generaciones a tratar y número de reinicios máximos (num_reinicios).

Al final de esta fase ya podíamos empezar a obtener los primeros resultados de aplicar el algoritmo CHC a la salida generada por el ensemble.

3.5 Fase de optimización del kNN en Matlab

Debido a que este algoritmo se utiliza constantemente para evaluar los descendentes generados por el algoritmo CHC fue necesario optimizar al máximo este código. Al principio propusimos dos métodos aunque al final probamos con hasta cuatro métodos distintos:

- **Repmat:** Realizar el algoritmo kNN aplicando la función interna de Matlab “repmat” para replicar la nueva instancia a clasificar tantas veces como instancias en el conjunto de train para así, a la hora de calcular la distancia entre dicha instancia y el conjunto entero no hiciese falta recorrer todo el dataset sino que se calcularía la distancia contra todo el dataset al mismo tiempo.
- **Bsxfun:** Realizar el algoritmo aplicando la función interna de Matlab “bsxfun” a la hora de realizar la resta entre la instancia a clasificar y cada una de las instancias del conjunto de train. Esta función principalmente lo que hace es aplicar entre distintos tamaños de matrices una función dada. Para ello, replica la matriz de menor tamaño, en nuestro caso la instancia a clasificar, tantas veces como la mayor, el conjunto de train, para después aplicar una función entre ellas de forma óptima, en nuestro caso la resta.
- **Combinación:** Realizar una combinación de los dos métodos anteriores de forma que, primero, replicábamos la nueva instancia a clasificar tantas veces como el número de instancias del conjunto de train con la función “repmat” para, después, aplicar la función “bsxfun” para realizar la resta entre las dos matrices.
- **Cubo:** Esta última opción es la utilizada en el proyecto. Es una opción que calcula la distancia entre todas las instancias a testear y todas las instancias del conjunto de train al mismo tiempo. Para ello, en vez de recibir como parámetro una instancia a clasificar, recibimos todo el conjunto de instancias a clasificar al mismo tiempo en forma de matriz. Con el conjunto de instancias a clasificar, lo que hacemos es replicar la matriz tantas veces como número de instancias contenidas en el conjunto de datos de train con la función “repmat”. Después, giramos el cubo intercambiando la primera y la tercera dimensión de forma que, visto de frente, en cada capa del cubo haya una instancia del conjunto de instancias a clasificar replicada tantas veces como instancias en el conjunto de train. De esta forma ahora es posible aplicar la función “bsxfun” entre cada una de las capas y el conjunto de train para obtener un

cubo con todas las distancias entre cada instancia del conjunto de train y el conjunto de test para la confianza obtenida por cada clasificador. De esta forma, solo falta realizar la suma de los valores del cubo en la segunda dimensión para tener una matriz que contiene las distancias entre todas las instancias. Proceso mostrado en la figura 9. Este método surge como solución al hecho de que en los anteriores existía un bucle para calcular la clase de cada una de las instancias del conjunto de test utilizando los anteriores métodos. En este, este bucle es suprimido y se realiza el cálculo de todas las clases de las instancias de test al mismo tiempo.

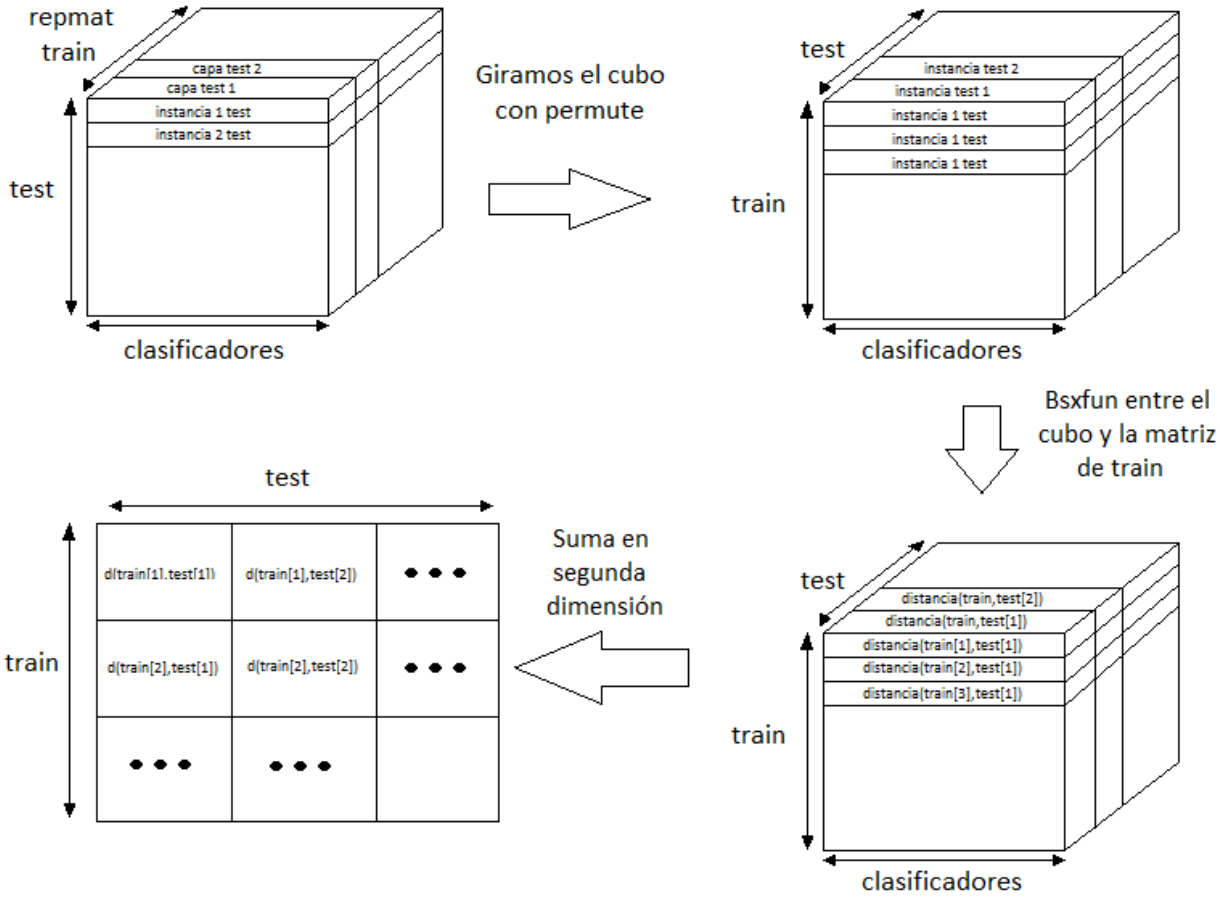


Figura 9 Cálculo de las distancias de kNN con el método del cubo

Tabla 2

(100 iter)	**Repmat**	**Bsxfun**	**Combinación**	**Cubo**
1ºEjecución	47.17s	36.43s	44.15s	42.24s
2ºEjecución	43.50s	39.23s	44.66s	42.50s

Tal y como se puede apreciar en la tabla 2, los resultados obtenidos con los cuatro métodos del algoritmo kNN con 100 ejecuciones, son relativamente parecidos. En orden de mejor a peor están así: Bsxfun, cubo, combinación y repmat. De esto podríamos deducir que el método más rápido es el método bsxfun. Aun así, en este proyecto utilizamos el método del cubo. Esto es debido a que el número de llamadas al algoritmo kNN desde el algoritmo CHC es muy alto. Además, en el algoritmo CHC el conjunto de datos de train y de test no varían durante la ejecución (en sí, los conjuntos de train y test son los mismos solo que es sobre el conjunto de train sobre el que seleccionamos instancias y características). Teniendo en cuenta esto, sabemos que el cubo generado no es necesario calcularlo para cada llamada del algoritmo kNN sino que con generarlo al inicio del CHC y pasarlo como parámetro al kNN es suficiente. Esto nos ahorra mucho tiempo en el cálculo del cubo por lo que es más eficiente para este proyecto.

3.6 Fase de optimización del código

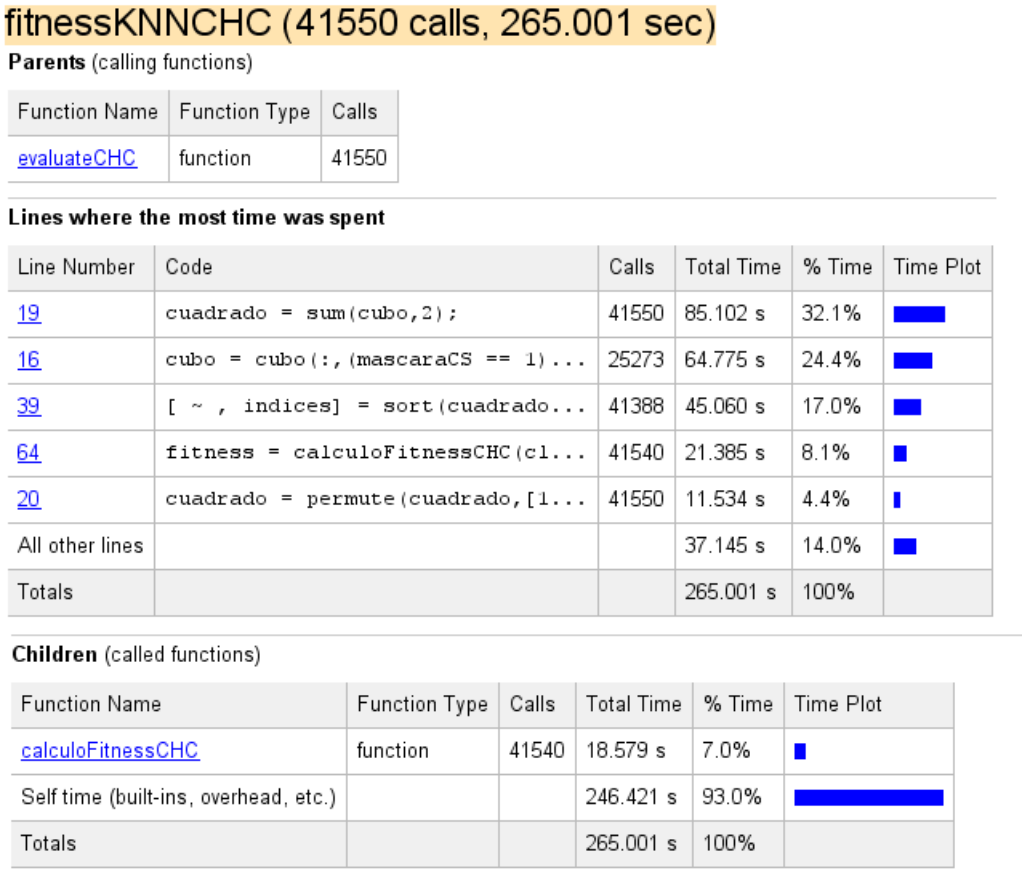
Debido a la necesidad de hacer una serie de pruebas finales con un coste computacional muy alto, tuvimos que optimizar el código completo. Para descubrir secciones costosas de código utilizamos la utilidad “Profiler” de Matlab, la cual destaca, tras una ejecución del programa, las secciones de código más costosas, así como el número de llamadas realizadas a las mismas. Podemos ver el resultado de una ejecución de ejemplo en las imágenes siguientes.

Profile Summary

Generated 24-Jun-2014 17:59:30 using cpu time.

Function Name	Calls	**Total Time**	Self Time*	Total Time Plot (dark band = self time)
main	1	279.201 s	0.375 s	
CHC	5	277.785 s	1.568 s	
evaluateCHC	3320	271.668 s	6.667 s	
fitnessKNNCHC	41550	265.001 s	246.421 s	
calculoFitnessCHC	41540	18.579 s	18.579 s	
HUXCHC	3299	3.212 s	3.212 s	
sortrows	3315	0.957 s	0.957 s	
cuboKNN	17	0.861 s	0.687 s	
kNNCHC	10	0.477 s	0.074 s	
rng	1	0.282 s	0.000 s	
rng>getCurrentType	1	0.235 s	0.204 s	
repmat	17	0.174 s	0.174 s	
kNN	2	0.172 s	0.047 s	
doOutput	14	0.111 s	0.111 s	
reiniciarCHC	16	0.038 s	0.038 s	

Figura 10 Resultado "Profiler" Matlab de tiempos y llamas a funciones



fitnessKNNCHC (41550 calls, 265.001 sec)

Parents (calling functions)

Function Name	Function Type	Calls
evaluateCHC	function	41550

Lines where the most time was spent

Line Number	Code	Calls	Total Time	% Time	Time Plot
19	`cuadrado = sum(cubo,2);`	41550	85.102 s	32.1%	
16	`cubo = cubo(:,(mascaraCS == 1)...`	25273	64.775 s	24.4%	
39	`[ ~ , indices] = sort(cuadrado...`	41388	45.060 s	17.0%	
64	`fitness = calculoFitnessCHC(cl...`	41540	21.385 s	8.1%	
20	`cuadrado = permute(cuadrado,[1...`	41550	11.534 s	4.4%	
All other lines			37.145 s	14.0%	
Totals			265.001 s	100%	

Children (called functions)

Function Name	Function Type	Calls	Total Time	% Time	Time Plot
calculoFitnessCHC	function	41540	18.579 s	7.0%	
Self time (built-ins, overhead, etc.)			246.421 s	93.0%	
Totals			265.001 s	100%	

Figura 11 Resultado "Profiler" Matlab de la función fitnessKNNCHC

Las optimizaciones más importantes descubiertas gracias al "Profiler" de Matlab son las siguientes:

- El uso de la función "find" a la hora de seleccionar las instancias a tener en cuenta en la función de evaluación. Se comprobó que trabajar de esa forma era lenta y costosa, además del gran número de llamadas que se realizaba a dicha función. Se sustituyó el trabajo con la función "find" por trabajar con los índices de las matrices y arrays, la cual es mucho más rápida.
- Debido a una segunda revisión con el "Profiler" de Matlab, se descubrió que aunque todos los campos contenidos en una máscara producida por el CHC estuviesen a 1, lo que significa que no hacía falta excluir ninguna instancia ni característica del conjunto, el trabajo con los índices costaba un tiempo notorio. Por ello se introdujeron dos comprobaciones para que en los casos en los que la máscara esté compuesta de todo valores a 1 evite realizar la fase de selección de instancias o características del conjunto.

A parte de estas optimizaciones más notorias, se han llevado a cabo otras más puntuales y menos importantes las cuales no merece la

pena mencionar ya que la mejora ha sido pequeña en comparación con lo anterior.

3.7 Fase de aclaración de código y últimos añadidos

Debido a las continuas modificaciones sufridas por el código hasta este punto, era muy difícil de entender y de modificar. Por ello, y por unos añadidos finales los cuales comentaremos más adelante, se decidió rehacer el código entero de forma mucho más clara, limpia y fácil para modificar.

Como añadidos, se propusieron dos tipos más de reducción de instancias y características. Estos añadidos surgieron de la necesidad de utilizar técnicas para conjuntos de datos no balanceados en la fase de la reducción con el algoritmo CHC. Esto es, aunque el ensemble haya sido construido con técnicas para conjuntos de datos no balanceados, el no utilizarlos en el CHC estaba perjudicando a los resultados obtenidos.

1. **MS (Majority Selection):** Este método reduce el número de instancias de la clase con mayor número de apariciones en el conjunto de entrenamiento. Con esto se pretende favorecer la aparición de instancias de la clase minoritaria, la que menos representación tenga en el conjunto. El objetivo de este método es el de mejorar el número de aciertos a la hora de clasificar la clase minoritaria.
2. **MS-FS (MS-CS para nosotros):** Este método es una combinación de los métodos MS y CS/FS vistos anteriormente por lo que, además de llevar a cabo una reducción del número de instancias de la clase mayoritaria, lleva a cabo una reducción del número de características que representan el conjunto de datos.

Este tipo de reducciones, una vez mejorado el código para adaptarse a nuevos cambios, no fueron complicadas. Solamente hubo que añadir dos tipos de inicializaciones más al CHC, añadir dos alternativas distintas a la hora de generar las máscaras tanto de IS como de CS y añadir dos formas más de generar los resultados basándose en las anteriores.

Otro de los añadidos finales fue el de, en vez de tener en cuenta como fitness del algoritmo CHC el resultado obtenido por o bien la media geométrica o bien la fmean o AUC, utilizar la reducción aplicada en cada cromosoma a través de una de dos fórmulas posibles [7]. La primera, a la

hora de calcular el fitness utilizar o incluir por medio de dos nuevos parámetros (*alfa* para las instancias y *beta* para las características) información sobre la reducción aplicada tanto en instancias como en características (9), lo cual favorece a los cromosomas que reduzcan tanto instancias como características. La segunda, utilizar información dependiendo tanto del número de instancias de la clase mayoritaria como minoritaria utilizando un nuevo parámetro para regular el peso de la diferencia de instancias de la clase mayoritaria y la minoritaria (10). Este caso surge de utilizar las reducciones MS y MS-CS los cuales priorizan la reducción de instancias de la clase mayoritaria.

(9)$Fitness(S) = \alpha * \beta * Medida + (1-\alpha) * InstanceReduction + (1-\beta) * FeatureReduction$, Donde Medida será o bien la Gmean, Fmean o AUC.

(10)$$Fitness(S) = \begin{cases} Medida - \left|1 - \frac{N^+}{n^-}\right| * P & if\ n^- > 0 \\ Medida - P & if\ n^- = 0 \end{cases}$$

Para tratar los resultados obtenidos por el programa a posteriori, tuvimos que modificar la forma de mostrar los resultados, de mostrarlos por pantalla, la forma utilizada durante la programación del programa completo para ir comprobando el correcto funcionamiento del mismo, a la generación de ficheros, los cuales trataremos en una fase final del proyecto. Estos ficheros guardan como matrices los resultados obtenidos en las ejecuciones, tanto en test como en train. Con los resultados nos referimos a los valores de TNrate, TPrate, Gmean, Fmean y AUC, comentados anteriormente, obtenidos como resultado. Además, en cada caso para los diferentes tipos de reducción aplicados se guardan los porcentajes de reducción, ya sea de instancias (diferenciando entre reducción de la clase mayoritaria y minoritaria), de características o de ambas.

Por último, se hizo una función main configurable para ciertos parámetros de entrada como, ruta de los ficheros a emplear, valor de k, reducción a aplicar y la función de fitness a utilizar.

3.8 Fase de preparación, ejecución de los lanzamientos y agregación de resultados

En esta fase, se llevaron a cabo varias preparaciones para el lanzamiento final del proyecto en el cluster y el posterior tratamiento de las salidas obtenidas.

La primera preparación fue la de crear, basándonos en otros casos, un fichero de Matlab con la función de generar la estructura de carpetas

que almacenaría tanto los resultados como los scripts de configuración de todas las combinaciones de la ejecución del ensemble, basándonos en la estructura de KEEL. Además, en esta preparación, también creamos un script para ejecutar de forma paralela en el cluster todas las combinaciones posibles del ensemble para generar como resultados los ficheros necesarios para la segunda fase.

La segunda fue la de crear una función de lanzamientos con Matlab. Esta función recibe como entrada los siguientes parámetros:

- Las rutas de los ficheros que contienen los conjuntos de train y test.
- Los ficheros que contienen los valores de alfa a aplicar a dichos conjuntos, los utilizados en boosting.
- El número del sub-conjunto del dataset tratado (i).
- El nombre del dataset a tener en cuenta, el cual también contiene la ruta completa al mismo, de la cual extraemos la técnica aplicada en el ensemble y el la cantidad de clasificadores utilizados en el mismo.

En definitiva, el objetivo de esta función es el de crear toda la estructura de carpetas para guardar los resultados obtenidos durante la ejecución del programa completo, generando toda la casuística para los datasets recibidos como parámetros y lanzando todas las combinaciones de la función main anteriormente preparada de forma parametrizable. En esta misma preparación, al igual que en la anterior, se tuvo que llevar a cabo otro script para ejecutar los lanzamientos de forma paralela en el cluster. En la imagen siguiente mostramos un ejemplo del script encargado de ejecutar los lanzamientos de forma paralela.

También, se programó una función de Matlab para tratar los resultados obtenidos tras la ejecución completa del proyecto. Esta función es la encargada de recorrer la estructura de carpetas generada como resultado, y de agregar los resultados obtenidos con las diferentes ejecuciones del programa, dando como resultado unos últimos ficheros que contengan los resultados obtenidos ante las diferentes combinaciones y los diferentes datasets tratados.

Por último, se llevó a cabo un estudio estadístico de los resultados obtenidos. Para ello, nos basamos en código anterior y en funciones ya programadas por lo que la mayor parte del trabajo fue generar las tablas de resultados de forma fácilmente legible.

4.- Marco experimental

En esta sección vamos a explicar el marco experimental tenido en cuenta en las ejecuciones de este proyecto. En la sección 4.1 explicaremos en detalle todos los parámetros tenidos en cuenta en el proyecto. La sección 4.2 explica la medida de evaluación elegida. La sección 4.3 presenta los test estadísticos realizados con los resultados de las ejecuciones. La sección 4.4 muestra los datasets empleados en las ejecuciones. La sección 4.5 presenta las diferentes configuraciones tenidas en cuenta en el ensemble. Por último, en la sección 4.6 se explica la combinatoria de parámetros realizada en la ejecución del proyecto.

4.1 Parámetros

Debido a la gran cantidad de parámetros tenidos en cuenta en el proyecto, vamos a tratar de explicar todos y remarcar a qué funciones pertenecen cada uno de ellos.

El primer grupo de parámetros son los recibidos en la función de lanzamientos, encargada de lanzar las ejecuciones con toda la combinatoria tratada.

- **outputTra:** Este parámetro recibe el conjunto de datos de entrenamiento generado a partir de las confianzas obtenidas en cada uno de los clasificadores C4.5 del ensemble.
- **outputTst:** Este parámetro es igual al anterior solo que recibe el conjunto de datos de test.
- **outputAlfaTra:** Este parámetro recibe el conjunto de alfas a aplicar en las técnicas de boosting del conjunto de train.
- **outputAlfaTst:** Este parámetro es igual al anterior solo que del conjunto de test.
- **i:** Este parámetro indica el número de partición a tratar.
- **dataName:** Este parámetro contiene tanto el nombre del data-set a tratar como la ruta para acceder al mismo, de la cual extraemos el tipo de ensemble y el número de clasificadores utilizados.

El siguiente grupo de parámetros pertenece a la función main, la cual ejecuta el proyecto con los parámetros especificados:

- **outputTra, outputTst, outputAlfaTra, outputAlfaTst, i:** Estos parámetros son los recibidos en la función de los lanzamientos.
- **K:** Su valor puede variar entre “ENSEMBLE”, lo cual indica que la ejecución que queremos realizar, o bien es calcular los resultados obtenidos en el ensemble, o bien es el ensemble aplicando el algoritmo CHC directamente sobre las salidas obtenidas sin agregar utilizando el kNN. En caso de tener un valor distinto a “ENSEMBLE”, se tratará como el parámetro pasado al algoritmo kNN como valor de k.
- **Dimensión:** Es el parámetro que indica la reducción a aplicar en el algoritmo CHC, con los siguientes valores:
  - **1:** Instance Selection
  - **2:** Feature/Clasifier Selection
  - **3:** Instance and Feature/Clasifier Selection
  - **4:** Majority Selection
  - **5:** Majority and Feature/Clasifier Selection
- **Fitness:** Este parámetro regula el tipo de fórmula utilizada en el cálculo del fitness, y los parámetros que esta conlleva:
  - **REDUCCION:** Aplica en el fitness la fórmula que da cierta prioridad a la reducción de instancias y/o de características, con los parámetros:
    - **alfaFitness:** 0.6
    - **betaFitness:** 0.99

    Estos parámetros han sido seleccionados en base a otras publicaciones en las que nos basamos para la realización de este proyecto.
  - **PRECISION:** No tiene en cuenta nada a parte del valor obtenido por la fórmula de fitness seleccionada, ya sea Gmean, Fmean o AUC.
  - **EBUS:** Aplica en el fitness la fórmula que da cierta prioridad a la reducción de instancias de la clase mayoritaria, en base a un parámetro:
    - **P:** 0.2

Además, existen varios parámetros ya fijados en el propio main, debido a la dificultad de ejecutar una combinatoria extremadamente amplia del problema: que esta conlleva:

- **tipoFitness:** Es el parámetro que regula el tipo de medida de evaluación a utilizar en el fitness. Está establecido a 1.
  - **1:** Gmean
  - **2:** Fmean
  - **3:** AUC

- **L:** Indica la longitud de la población a tener en cuenta en el algoritmo genético. En este caso su valor es 50.
- **num_iteraciones:** Indica el número máximo de iteraciones a realizar en el algoritmo genético. En este caso su valor es 10000.
- **num_reinicios:** Indica el número máximo de reinicios a realizar en el algoritmo genético. Está establecido en el proyecto a 3.
- **prob0to1:** Indica la probabilidad de cambiar un bit de 0 a 1 en el cruce HUX. Basándonos en otras publicaciones hemos establecido su valor a 0.25.
- **Semilla:** Indica la semilla utilizada para generar los números aleatorios. Sirve para que siempre se utilice la misma y que los resultados no se vean afectados por semillas distintas. Se utiliza la semilla 15240.

4.2 Medidas de evaluación

Debido a la inmensa combinatoria contemplada en la ejecución del proyecto, ha sido necesaria una reducción de la misma. Por ello, de las tres medidas de evaluación contempladas y programadas en la ejecución del proyecto solamente se ha tenido en cuenta una de ellas, la media geométrica. Para entender el porqué de esta elección vamos a empezar por explicar cómo es cada una de ellas.

- La media geométrica, por ser una multiplicación entre el TNrate y el TPrate, si una de ellas es nula, ambas son nulas. Esto promueve que se clasifiquen bien instancias de las dos clases.
- La Fmean, se basa en calcular la precisión (porcentaje de clasificadas correctamente como positivas entre todas las clasificadas como positivas) y el recall (porcentaje de positivas clasificadas positivas correctamente). Aplicando esto, se logra tener en cuenta los fallos realizados en la clasificación de las positivas, aunque no tiene muy en cuenta los aciertos obtenidos en la clase negativa.
- La AUC, se basa en hacer una media entre el TNrate y el TPrate. Esta medida no es muy útil ya que podríamos obtener un fitness de 0.5 con acertar todas las instancias de una clase y ninguna de la otra, lo cual pretendemos evitar a toda costa.

Teniendo en cuenta estas tres medidas, elegimos utilizar la Gmean debido a que es muy útil el hecho de que si no se acierta en la clasificación de ninguna instancia de una clase devuelve un 0. Esto para el caso de los conjuntos de datos no balanceados es muy importante ya que evita la tendencia de acertar la mayoría de la clase mayoritaria y dejar de lado las instancias de la clase minoritaria.

4.3 Test estadísticos

En este proyecto se lleva a cabo un estudio estadístico para determinar si la nueva propuesta mejora la fase de agregación del ensemble clásico o no. Para ello, se llevan a cabo dos tipos de test estadísticos, el test de Wilcoxon y el test de Friedman.

El test de Wilcoxon sirve para determinar la independencia estadística entre dos métodos dados. Para ello tiene en cuenta los resultados obtenidos en los diferentes conjuntos de datos utilizados en la ejecución. Cuantos más data-sets utilizados más preciso será el resultado del test. Para realizar este método calcula una serie de rangos con los que más adelante trabajaremos para determinar cuál de los dos métodos propuestos es el mejor. La independencia estadística queda marcada por un p_{valor}, el cual se obtiene en base a los rangos. Si el p_{valor} es menor que 0,05 entonces el método con más rangos es estadísticamente mejor que el de menor número de rangos. En caso contrario, no se puede afirmar que el método con mayor número de rangos sea estadísticamente mejor.

El test de Friedman sirve para determinar la independencia estadística entre más de dos métodos distintos. Para ellos, al igual que el anterior, tiene en cuenta los resultados obtenidos en los diferentes conjuntos de datos utilizados en la ejecución. Por ello, cuantos más data-sets utilizados haya, más preciso será el resultado obtenido. Al igual que en el método anterior, para realizar este método se calculan una serie de rangos con los que trabajaremos. La independencia estadística queda marcada por un p_{valor}, al igual que en el caso anterior. Si dicho p_{valor} es menor que 0.05 entonces, al contrario que en el caso anterior, el método con menor número de rangos es estadísticamente mejor que el de menor número de rangos. En caso contrario, no se puede afirmar que el método con menor número de rangos sea estadísticamente mejor.

4.4 Datasets

Debido a que es necesario determinar si el método propuesto para mejorar la fase de agregación del ensemble clásico es en realidad mejor o no, es necesario realizar los test estadísticos. Estos, además, para que sean más precisos y más confiables, es necesario que cuenten con un número importante de ejecuciones sobre distintos datasets. Para ello, hemos utilizado los datasets mostrados en la siguiente tabla, la cual muestra qué contiene cada uno de ellos. Debido a que trabajamos con clasificadores binarios en todo momento y tratamos el problema de los

conjuntos de datos no balanceados, todos los datasets incluidos constan de dos clases y son no balanceados.

Tabla 3

Nombre	Atributos	Nº de Ejemplos
cleveland-0_vs_4	13	117
ecoli-0-1-3-7_vs_2-6	7	281
ecoli-0-1-4-6_vs_5	6	280
ecoli-0-1-4-7_vs_2-3-5-6	7	336
ecoli-0-1-4-7_vs_5-6	6	332
ecoli-0-1_vs_2-3-5	7	244
ecoli-0-1_vs_5	6	240
ecoli-0-2-3-4_vs_5	7	202
ecoli-0-2-6-7_vs_3-5	7	224
ecoli-0-3-4-6_vs_5	7	205
ecoli-0-3-4-7_vs_5-6	7	257
ecoli-0-3-4_vs_5	7	200
ecoli-0-4-6_vs_5	6	203
ecoli-0-6-7_vs_3-5	7	222
ecoli-0-6-7_vs_5	6	220
ecoli-0_vs_1	7	220
ecoli1	7	336
ecoli2	7	336
ecoli3	7	336
ecoli4	7	336
glass-0-1-2-3_vs_4-5-6	9	214
glass-0-1-4-6_vs_2	9	205
glass-0-1-5_vs_2	9	172
glass-0-1-6_vs_2	9	192
glass-0-1-6_vs_5	9	184
glass-0-4_vs_5	9	92
glass-0-6_vs_5	9	118
glass0	9	214
glass1	9	214
glass2	9	214
glass4	9	214
glass5	9	214
glass6	9	214
haberman	3	306
iris0	4	150
led7digit-0-2-4-5-6-7-8-9_vs_1	7	443
new-thyroid1	5	215
new-thyroid1	5	215
shuttle-c2-vs-c4	9	129
yeast-1_vs_7	7	459
yeast-2_vs_4	8	514
yeast-2_vs_8	8	482

4.5 Configuraciones del ensemble

Debido a las diferentes configuraciones tenidas en cuenta en el proyecto, es difícil seguir todas. Hasta ahora hemos explicado los parámetros tenidos en cuenta en el proyecto pero no claramente los tenidos en cuenta en el ensemble. Por ello vamos a proceder a explicar la configuración del ensemble.

- El algoritmo base sobre cada clasificador es el algoritmo C4.5 en todo momento.
- La semilla empleada para generar los números aleatorios es 24123124.
- La poda pesimista de árboles, la cual explicamos en la sección 2.2.1, está en todo momento activada.
- El número de clasificadores establecido depende del algoritmo empleado. Cada uno de ellos se prueba con dos cantidades distintas.
    - **RUSBOOST:** 10 o 40.
    - **SMOTEBAGGING:** 40 o 100.
    - **UNDERBAGGING:** 40 o 100.
    - **EASYENSEMBLE:** Este caso es algo distinto ya que siempre utilizamos 10 clasificadores como parámetro pero variamos el número de “bolsas” a utilizar, las cuales contienen cada una 10 clasificadores. Los dos tamaños de estas “bolsas” son 4 y 10, por lo que en total tendrán 40 y 100 clasificadores.
- El número mínimo de instancias por hoja está establecido a 2.
- El método de entrenamiento en todo momento está establecido a NORESAMPLING, lo que significa que no hay reemplazamiento de instancias una vez utilizadas.
- El porcentaje de balanceo entre la clase mayoritaria y la minoritaria para el algoritmo SMOTE está establecido al 50%.

4.6 Combinatoria de parámetros

Debido a la gran combinatoria de parámetros tenida en cuenta en la ejecución del proyecto vamos a tratar de hacerla más comprensible. En primer lugar está el tipo de ensemble creado.

- EASYENSEMBLE
- RUSBOOST
- UNDERBAGGING
- SMOTEBAGGING

En segundo lugar está el número de clasificadores que vamos a utilizar, el cual varía dependiendo del tipo de ensemble empleado. Después, está el valor de k aplicado o para los casos en los que no se aplica el algoritmo kNN la palabra “ENSEMBLE” para diferenciarlos. Este valor varía entre 1, 3 y 5.

Partiendo de los métodos en los que empleamos el algoritmo kNN, el siguiente nivel es la dimensión aplicada (IS, CS, IS-CS, MS o MS-CS) en caso de aplicar el algoritmo CHC o la palabra “kNN” para diferenciarlos en caso contrario. En el caso de kNN, la última configuración es el dataset utilizado en la ejecución. En caso contrario, el siguiente parámetro a tener en cuenta es el tipo de fórmula utilizada para calcular el fitness, EBUS, PRECISSION o REDUCCION (para el caso de CS el EBUS no tiene sentido utilizarlo ya que no se llevan a cabo reducciones de instancias por lo que lo descartamos). Por último por este camino, tenemos el dataset utilizado en la ejecución del proyecto.

Siguiendo por el caso de que k en vez de contener un valor numérico contenga “ENSEMBLE”, tenemos dos posibles casos. “ORIGINAL” es el caso en el que se calculan los resultados obtenidos por el ensemble original para cada dataset. “CS” es la reducción aplicada en este caso tras aplicar a la salida obtenida en el ensemble el algoritmo CHC. En este caso solo existen dos tipos de fórmulas utilizadas después, PRECISION y REDUCCION, ya que como hemos dicho antes no tiene sentido aplicar EBUS. Una vez aquí, solamente depende del dataset utilizado.

Para aclarar un poco esta explicación se muestra una estructura de árbol en la siguiente imagen, la cual es la seguida en la combinatoria de los parámetros utilizados.

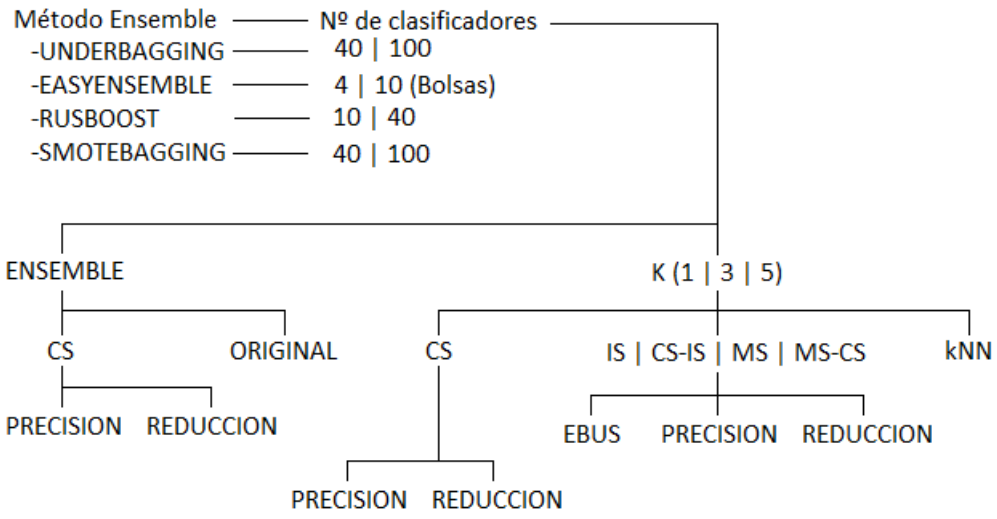


Figura 12 Árbol de combinatoria de parámetros utilizados

En la figura anterior, falta decir que tras cada uno de los niveles “finales” u hojas del árbol, están los diferentes datasets utilizados ya que las pruebas las hacemos con todos ellos.

En total, el número de ejecuciones es la siguiente:

$$Nº\ Ejecuciones = (Métodos\ Ensemble)4 *$$

$$(Cantidades\ de\ clasificadores)2 *$$

$$\begin{pmatrix}(Ensemble + CHC)2 * DATA + (Ensemble)1 * DATA + \\ (E + kNN)3 * DATA + (E + CHC + kNN)\ 3 * x * DATA\end{pmatrix}$$

Donde DATA es el número de datasets tratados y x es:

$$x = (CS)2 + (Resto\ de\ reducciones)4 * 3$$

Siendo DATA 42, el total de ejecuciones realizadas es:

$$Nº\ Ejecuciones = 4 * 2 * (2 * 42 + 42 + 3 * 42 + 3 * (2 + 4 * 3) * 42)$$
$$= 16128\ Ejecuciones$$

Para ser más precisos, hay que destacar que cada dataset está particionado en 5 por lo que el valor DATA en vez de 42 es 210. Por tanto:

$$Nº\ Ejecuciones\ = 80640\ Ejecuciones$$

5.- Estudio experimental

En este capítulo, vamos a exponer y explicar los resultados obtenidos en la ejecución del proyecto. En la sección 5.1 exponemos una introducción y los objetivos del proyecto. En la sección 5.2 explicamos el significado de la figura mostrada como resultado. En la sección 5.3 presentamos el modelo seguido en las comparaciones de los diferentes métodos. Finalmente, en la sección 5.4 exponemos y comentamos los resultados obtenidos.

5.1 Introducción y objetivos

El objetivo de este proyecto es mejorar la fase de agregación de los ensembles proponiendo nuevos métodos para ello. Los nuevos métodos propuestos son muchos pero se pueden reducir a tres familias distintas:

1. Utilizar el algoritmo kNN para agregar la salida del ensemble (En las figuras y tablas “E+KNN”).
2. Utilizar el algoritmo kNN para agregar la salida del ensemble tras utilizar el algoritmo CHC para reducir el número de instancias y de características del conjunto de datos de entrenamiento obtenidos como salidas del ensemble (En las figuras o tablas “E+KNN+CHC”).
3. Aplicar a la salida del ensemble el algoritmo CHC para reducir el número de clasificadores utilizados en el ensemble y después agregar los resultados con el voto (En las figuras o tablas “E+CHC”). Este último caso no varía la forma habitual de agregar los resultados del ensemble original pero sí modifica las salidas obtenidas en el ensemble por lo que lo incluimos como una opción.

Cada una de estas familias incluye diferentes métodos, dependiendo de todos los parámetros comentados con anterioridad. En total, son 47 métodos distintos, 3 en la primera familia (Dependiendo del valor de k), 42 en la segunda familia (Dependiendo del fitness, el valor de k y la reducción aplicada) y 2 en la tercera familia (Dependiendo del fitness utilizado).

Debido a la gran cantidad de métodos propuestos, es necesaria una comparación reducida de todos ellos, no podemos comparar todos

con todos de golpe. Para ello, vamos a llevar a cabo una comparación hecha por niveles. Solo los mejores de cada nivel serán comparados en el siguiente nivel.

Tal y como hemos comentado, a la hora de realizar las comparaciones vamos a utilizar dos tipos de test estadísticos, el test de Wilcoxon y el test de Friedman.

- **Wilcoxon:** Utilizado en los casos en los que la comparación sea entre dos métodos distintos. En este caso el método con mayor número de rangos es estadísticamente mejor si el p_{valor} es menor que 0.05. En caso contrario no podemos afirmar que el método con mayor número de rangos sea estadísticamente mejor.
- **Friedman:** Utilizado para comparar tres o más métodos distintos. En este caso el método con menor número de rangos es estadísticamente mejor en el caso de que el p_{valor} sea menor que 0.05. En caso contrario no podemos afirmar que el método con menor número de rangos sea estadísticamente mejor.

5.2 Figura de resultados

En este apartado vamos a explicar qué muestra la figura 16, la cual contiene un resumen de los resultados obtenidos en el proyecto con el método de UnderBagging con 40 clasificadores (resultados en la figura 13). En ella se muestran los resultados de las comparativas entre los diferentes métodos, los propuestos y el original, para la fase de agregación (excluyendo la familia de “Ensemble + CHC”) del ensemble. Estos test estadísticos han sido calculados con los resultados obtenidos con la media geométrica o Gmean.

Para empezar, en el eje vertical tenemos cinco columnas. La primera indica la familia de métodos a la que pertenece el resultado a mostrar, y al principio de la segunda indica la configuración de cada método. Las familias son las siguientes:

- **ENSEMBLE:** Hace referencia a los resultados obtenidos utilizando el ensemble, en este caso UnderBagging con 40 clasificadores, con una fase de agregación basada en el voto.
- **ENSENBLE + CHC:** Hace referencia a la familia de resultados obtenidos tras aplicar el algoritmo CHC para la reducción de clasificadores que componen el ensemble, para después agregar los resultados con el voto.

	ENSEMBLE + CHC + KNN														
	k = 1														
	IS			CS		IS-CS			MS			MS-CS			
DATASET	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	EBUS	
cleveland-0_vs_4	0,7926952	0,6183979	0,7143374	0,7104649	0,6761639	0,6220635	0,7172380	0,8550514	0,7651959	0,7475821	0,7755602	0,8027181	0,8118949	0,7469529	
ecoli-0-1-3-7_vs_2-6	0,7314163	0,6778811	0,6765776	0,5358903	0,5377513	0,7369997	0,6797246	0,6715217	0,8570328	0,6765776	**0,8570328**	0,6709401	0,5377513	0,6709401	
ecoli-0-1-4-6_vs_5	0,8722428	0,8518181	0,8766218	0,8264583	0,8298219	0,8258411	0,8281318	0,8244747	0,8828631	0,8481533	0,8731765	0,8808138	0,8821924	0,8214213	
ecoli-0-1-4-7_vs_2-3-5-6	0,8482908	0,8366064	0,8718329	0,7930401	0,7970868	0,8269664	0,8316921	0,8338016	0,8486452	0,8719631	0,8584547	0,7835371	0,8342165	0,7890768	
ecoli-0-1-4-7_vs_5-6	0,8625467	0,8463867	0,8417986	0,7765890	0,7991067	0,8521974	0,8361674	0,8701100	0,8559928	0,8257979	0,8401027	0,8698964	0,8508412	0,8127649	
ecoli-0-1_vs_2-3-5	0,8059355	0,7535771	0,8403470	0,7596927	0,7357266	0,7427623	0,7756026	0,8387311	0,8108982	0,7642975	0,8014313	0,7973230	0,7680017	0,8077691	
ecoli-0-1_vs_5	0,8971515	0,8756551	0,9041465	0,7538028	0,7570543	0,8560573	0,7796374	0,8090783	0,9198898	0,9044291	0,8983713	0,8481362	0,9084391	0,8524560	
ecoli-0-2-3-4_vs_5	0,8598864	0,8356414	0,8520017	0,8091003	0,8746860	0,8063836	0,8225802	0,8477428	**0,9260763**	0,8928119	0,9079267	0,7292233	0,8485289	0,8379779	
ecoli-0-2-6-7_vs_3-5	0,8553726	0,7375879	0,8041997	0,7997233	0,7809800	0,7561027	0,7618508	0,8170206	0,7974609	0,7078464	0,7695382	0,8179977	0,8192718	0,7573769	
ecoli-0-3-4-6_vs_5	0,8728241	0,8508099	0,8484532	0,8397643	0,8802369	0,8993846	0,8823510	0,8682819	0,9148782	0,9042329	0,8991220	0,9123027	0,8441356	0,9227449	
ecoli-0-3-4-7_vs_5-6	0,8997361	0,8375008	0,8818925	0,7917690	0,8395195	0,8558774	0,8138603	0,8523724	0,8652857	0,8320621	0,8718464	0,8782737	0,8414328	0,8781000	
ecoli-0-3-4_vs_5	0,8199010	0,8536628	0,8536628	0,8243359	0,8211649	0,7618382	0,7910621	0,8318191	0,9171234	0,8780346	0,8781439	0,7960793	0,7955999	0,8566360	
ecoli-0-4-6_vs_5	0,9257231	0,8719198	0,9277011	0,9063486	0,8775997	0,8700268	0,8462707	0,8984718	0,9462061	0,9276994	**0,9490465**	0,8699235	0,8173834	0,8776099	
ecoli-0-6-7_vs_3-5	0,8323790	0,8140724	0,8331484	0,8007441	0,8033262	0,8306001	0,7813762	0,8399548	0,8748114	0,8659880	0,8561025	0,8521566	0,7969913	0,7989652	
ecoli-0-6-7_vs_5	0,8226328	0,7736956	0,7976052	0,7727628	0,7740208	0,8248616	0,8004610	0,8274099	**0,8840766**	0,7639809	0,7621843	0,8213918	0,8028153	0,8222795	
ecoli-0_vs_1	0,9731766	0,9766551	0,9766551	0,9766551	0,9766551	0,9803264	0,9766551	0,9803264	0,8418115	0,8452900	0,8355285	0,8418115	0,8452900	0,8489613	
ecoli1	0,8456367	0,8299460	0,8333153	0,8108069	0,8350189	0,8383438	0,8281867	0,8382664	0,8526284	0,8376011	0,8352432	0,8494695	0,8178142	0,8436108	
ecoli2	0,8916789	0,9132354	0,9084247	0,8690941	0,8855906	0,9016979	0,8875139	0,9004792	0,8980992	0,9047438	0,8911743	0,8971030	0,8858162	0,8997693	
ecoli3	0,8441424	0,7997856	0,8123494	0,7810587	0,7980595	0,8230399	0,8044695	0,8405283	0,8013482	0,8042387	0,8014841	0,8296528	0,8124446	0,8013478	
ecoli4	0,9060799	0,8489929	0,8819056	0,8099637	0,8843138	0,9053026	0,9068963	0,9354919	0,8740700	0,8807766	0,9097394	0,9079158	0,9093330	0,9109337	
glass-0-1-2-3_vs_4-5-6	0,9081092	0,9050710	0,9114409	0,8869752	0,9038010	0,8967889	0,9074895	0,9009954	0,9235666	0,9144464	0,9114409	0,9170518	0,8901251	0,8838722	
glass-0-1-4-6_vs_2	0,5802267	0,4435191	0,5564640	0,4108092	0,4108092	0,5494062	0,4154830	0,4406786	0,7151677	0,4379363	0,6472288	0,5527111	0,5208892	0,5555515	
glass-0-1-5_vs_2	0,6949755	0,6955434	0,6520637	0,5159169	0,5212541	0,6047326	0,5639145	0,6476848	0,6576051	0,6817505	0,6676104	0,5112091	0,6555628	0,5568666	
glass-0-1-6_vs_2	0,5419164	0,4997001	0,5414569	0,2490332	0,2123696	0,4014167	0,4631771	0,4971755	0,6243642	0,5264175	0,5850236	0,4292693	0,5424985	0,5920605	
glass-0-1-6_vs_5	0,9764878	0,7822862	**0,9764878**	0,7942443	0,7942443	0,7942443	0,7942443	0,7942443	0,9677244	0,7735227	0,7735227	0,7942443	0,7942443	0,7942443	
glass-0-4_vs_5	**0,9936492**	0,9350705	**0,9936492**	**0,9936492**	0,9350705	**0,9936492**	0,9350705	**0,9936492**	**0,9936492**	0,9350705	**0,9936492**	**0,9936492**	0,9350705	**0,9936492**	
glass-0-6_vs_5	**0,9949359**	0,9363572	**0,9949359**	0,9363572	0,9363572	0,9258122	0,9363572	0,9363572	**0,9949359**	0,9363572	**0,9949359**	0,9363572	0,9363572	0,9363572	
glass0	0,8125976	0,8049162	0,8026839	0,7741065	0,7388130	0,7580838	0,7642424	0,7731149	0,8164337	0,7984713	0,8200707	0,7857361	0,7576333	0,7817147	
glass1	0,7708122	0,7273545	0,7411860	0,7222692	0,7137948	0,7512293	0,7479080	0,7472434	0,7966847	0,7949261	0,7754913	0,7536372	0,7793056	0,7460130	
glass2	0,4550852	0,4615232	0,3477787	0,4739143	0,3127221	0,3477787	0,3536225	0,5998766	0,5342506	0,5044155	0,4959104	0,4226604	0,3518205	0,4660928	
glass4	0,7078874	0,7512745	0,7441264	0,6817569	0,6838111	0,7098988	0,7175513	0,7104032	0,7032217	0,7099681	0,7415414	0,7027540	0,7491926	0,7032309	
glass5	0,9290892	0,7365132	0,7365132	0,7365132	0,7365132	0,7365132	0,7365132	0,7365132	0,9290892	0,7365132	0,7365132	0,7365132	0,7365132	0,7365132	
glass6	0,9151774	0,9176616	0,9176616	0,9347550	0,8960366	0,9344748	0,9228096	0,9177704	0,9047497	0,9097889	0,9072338	0,9151774	0,9289943	0,9177704	
haberman	0,5713559	0,5857341	0,5699565	0,5936207	0,5905129	0,6248341	0,6066508	0,6046872	0,5904161	0,5869959	0,5911199	0,5659478	0,5690608	0,5997017	
iris0	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	
led7digit-0-2-4-5-6-7-8-	0,8170889	0,8153096	0,8115296	0,8019707	0,8142851	0,8198087	0,8193175	0,8093076	0,7886826	0,7942611	0,7885517	0,7874375	0,7885907	0,7886914	
new-thyroid1	0,9621033	0,9488077	0,9488077	0,9488077	0,9488077	0,9309626	0,9488077	0,9427535	**0,9859328**	0,9605986	0,9582344	0,9464435	0,9488077	0,9464435	
new-thyroid1	0,9711371	0,9739747	0,9565086	0,9501049	0,9473075	0,9444699	0,9498974	0,9473075	0,9711371	0,9711371	0,9711371	0,9472673	0,9473075	0,9444699	
shuttle-c2-vs-c4	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	
yeast-1_vs_7	0,6198925	0,5635762	0,6445756	0,4925868	0,5226740	0,6301450	0,5101874	0,5877441	0,7135815	0,6816094	0,7203106	0,5079744	0,5103793	0,5501373	
yeast-2_vs_4	0,8505142	0,8561153	0,8701858	0,8723903	0,8734416	0,9013701	0,8550071	0,8832173	0,9122643	0,8779920	0,9093715	0,8708395	0,8615258	0,8907907	
yeast-2_vs_8	0,7588196	0,7962153	0,7607382	0,7341097	0,7325639	0,7333473	0,7335205	0,7300603	**0,8114998**	0,7601246	0,7461627	0,7283309	0,7622712	0,7294636	
MEDIA	**0,8328820**	**0,7983354**	**0,8191777**	**0,7724213**	**0,7709240**	**0,7975090**	**0,7862675**	**0,8136061**	**0,8490259**	**0,8087178**	**0,8284763**	**0,7964670**	**0,7949067**	**0,8014539**	

Figura 13 Tabla resultados UnderBagging con 40 clasificadores

Continuación tabla de resultados

		ENSEMBLE + CHC + KNN														
		k = 3														
		IS			CS		IS-CS			MS			MS-CS			
DATASET	EBUS	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	EBUS	
cleveland-0_vs_4	0,7469529	**0,9235516**	0,8047514	0,7834424	0,7257562	0,7195377	0,7617899	0,7555650	0,7026046	0,8532051	0,7087585	0,8693917	0,7584188	0,7604063	0,7668432	
ecoli-0-1-3-7_vs_2-6	0,6709401	0,7314163	0,6797246	0,7351562	0,5377513	0,5414214	0,7295188	0,7351562	0,7319793	0,6709771	0,6778811	0,6619234	0,6734007	0,6778811	0,6657005	
ecoli-0-1-4-6_vs_5	0,8214213	0,8959924	0,8498433	0,9014360	0,8281318	0,8508945	0,9073164	0,8576248	0,8216841	0,9413767	0,8752554	0,9244600	0,8370770	0,9067320	0,8504422	
ecoli-0-1-4-7_vs_2-3-5-6	0,7890768	0,8705047	0,8599666	0,8732670	0,7919375	0,8173604	0,8508044	0,8571068	0,8530747	0,8712165	0,8613062	0,8567528	0,8065144	0,8454872	0,8558171	
ecoli-0-1-4-7_vs_5-6	0,8127649	0,8595641	0,8076498	0,8906913	0,7912263	0,7736321	0,8695136	0,7768193	0,8501569	0,8864295	0,8046406	0,8592557	0,8657481	0,7931920	0,8423790	
ecoli-0-1_vs_2-3-5	0,8077691	0,8711574	0,8357931	0,8736897	0,7959927	0,8077238	0,7430867	0,7838646	0,7710661	0,8902281	0,8268361	0,8811053	0,8517895	0,7858055	0,8605088	
ecoli-0-1_vs_5	0,8524560	0,9020602	0,8620012	0,8753176	0,8600636	0,7796374	0,7912180	0,8548220	0,8491460	0,9482923	0,8733149	0,8716797	0,8154285	0,7857156	0,8222709	
ecoli-0-2-3-4_vs_5	0,8379779	0,8834269	0,8092213	0,8646031	0,8610366	0,8071076	0,7991126	0,8481753	0,8054128	0,8834269	0,8669927	0,8669927	0,8625548	0,8433347	0,8366272	
ecoli-0-2-6-7_vs_3-5	0,7573769	0,8502422	0,7990999	0,8527905	0,7771542	0,8192718	0,8150709	0,8576239	0,8130954	0,8375977	0,7979802	0,8550756	0,8464907	0,8144723	**0,8708539**	
ecoli-0-3-4-6_vs_5	0,9227449	0,8966981	0,8456542	0,8994866	0,8589225	0,8853529	0,8858354	0,8752481	0,8483859	**0,9356347**	0,8908972	0,8673039	0,8602944	0,8799944	0,8678434	
ecoli-0-3-4-7_vs_5-6	0,8781000	0,8995287	0,8355875	0,8956215	0,8157351	0,8138185	0,8761112	0,8360360	0,8840316	0,8873880	0,8610038	0,8718918	0,8937141	0,8806911	0,8508302	
ecoli-0-3-4_vs_5	0,8566360	0,8412047	0,8536628	0,8508654	0,8252968	0,8268303	0,8855526	0,8561566	0,8252968	0,8622877	0,8508654	0,8456053	0,8441146	0,7931774	0,7911994	
ecoli-0-4-6_vs_5	0,8776099	0,9257231	0,8454894	0,9257231	0,8196227	0,8414571	0,8439060	0,8730576	0,9220616	0,9097596	0,8635211	0,8587838	0,8306202	0,8692111	0,8514438	
ecoli-0-6-7_vs_3-5	0,7989652	0,8595383	0,8668417	0,8668417	0,8303275	0,8312522	**0,8888049**	0,8594004	0,8591941	0,8682815	0,8264573	0,8525143	0,8495394	0,8576215	0,8231822	
ecoli-0-6-7_vs_5	0,8222795	0,8626521	0,7976052	0,8670858	0,8014613	0,7809884	0,8196975	0,7976052	0,8183544	0,8463864	0,7975538	0,8215830	0,8183713	0,7784564	0,8144631	
ecoli-0_vs_1	0,8489613	0,9804517	0,9804517	0,9804517	0,9803264	0,9803264	0,9766551	0,9803264	0,9766551	0,9694340	0,9731766	0,9731766	0,9694340	0,9768478	0,9731766	
ecoli1	0,8436108	0,8675376	0,8403479	0,8605183	0,8019201	0,8272521	0,8455787	0,8436249	0,8550755	0,8687583	0,8466585	0,8777042	0,8783550	0,8449751	0,8453426	
ecoli2	0,8997693	0,9011075	0,9255813	0,9029127	0,8814597	0,8916228	0,9039156	0,8933965	0,8831959	0,9185009	0,9222461	0,9171930	0,8979676	0,8952920	0,9066815	
ecoli3	0,8013478	0,8565526	0,7997781	0,8408224	0,8137899	0,8154089	0,8375988	0,8168309	0,8213194	0,8186436	0,8036639	0,8063615	0,8157375	0,8295624	0,8222459	
ecoli4	0,9109337	0,9377345	0,9152830	0,9391149	0,8815755	0,8251738	0,8789297	0,8553242	0,9107134	0,9345343	0,8569576	0,9391020	0,9322661	0,8857386	0,8778028	
glass-0-1-2-3_vs_4-5-6	0,8838722	0,9294686	0,9385856	0,9386651	0,8834299	0,8965553	0,9145907	0,8858697	0,8876643	0,9278143	0,9349980	0,9289754	0,9015100	0,8909676	0,9231358	
glass-0-1-4-6_vs_2	0,5555515	0,6069539	0,4119354	0,6464128	0,4154830	0,5277889	0,5393540	0,4096161	0,6463904	0,6232290	0,6537781	0,5536236	0,6359116	0,4435191	0,5406966	
glass-0-1-5_vs_2	0,5568666	**0,7191816**	0,6534725	0,6736179	0,4991946	0,5771619	0,6011080	0,5616069	0,5477728	0,6585872	0,6372024	0,6669499	0,6369001	0,6527379	0,6413501	
glass-0-1-6_vs_2	0,5920605	0,3850705	0,5053131	0,5509483	0,3478866	0,3075113	0,5328059	0,3556658	0,5670324	0,6345550	0,5758803	0,5567570	0,5281771	0,4031286	0,4014167	
glass-0-1-6_vs_5	0,7942443	0,9736100	0,7794083	0,9736100	0,7942443	0,7942443	0,7942443	0,7942443	0,7942443	0,9736100	0,7794083	0,9736100	0,7942443	0,7942443	0,7942443	
glass-0-4_vs_5	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	
glass-0-6_vs_5	0,9363572	**0,9949359**	**0,9949359**	**0,9949359**	0,9363572	0,9363572	0,9843909	**0,9949359**	0,9363572	**0,9949359**	**0,9949359**	**0,9949359**	0,9363572	0,9363572	0,9258122	
glass0	0,7817147	0,8482346	0,8127282	0,8224395	0,7746155	0,7534798	0,8263780	0,7945510	0,7855193	0,8262228	0,8224792	**0,8549657**	0,7786067	0,7768472	0,8066909	
glass1	0,7460130	0,7746454	0,7860349	0,7444423	0,7087595	0,7104865	0,7836867	0,7559461	0,7630439	0,7777287	0,7670010	0,7751210	0,7655619	0,7542645	0,7831565	
glass2	0,4660928	0,6082547	0,4630330	0,4496228	0,3469547	0,3514848	0,4598927	0,4254443	0,4611669	0,4481271	0,4587188	0,4539958	0,5737117	0,4593831	0,4947383	
glass4	0,7032309	0,7441264	0,7200249	0,7415414	0,7078874	0,7154706	0,7031941	0,7537903	0,7036293	0,7031941	0,7150343	0,7031941	0,7053023	0,7150355	0,7006458	
glass5	0,7365132	0,9876679	0,7365132	0,9876679	0,7365132	0,7365132	0,9876679	0,9365132	0,9926069	0,9876679	0,7365132	0,9290892	0,9290892	0,7365132	0,7340282	
glass6	0,9177704	0,9207512	0,9151427	0,9151427	0,9177704	0,9203255	0,9292348	0,9177704	**0,9372738**	0,9046230	0,9151427	0,9023083	0,9266796	0,9150492	0,9203255	
haberman	0,5997017	0,6205624	0,5858698	0,6299585	0,6054801	0,5970553	0,6267658	0,6135664	0,6291997	0,6079632	0,6052782	0,6175981	0,6050215	0,6044381	0,6109924	
iris0	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	
led7digit-0-2-4-5-6-7-8-	0,7886914	0,8028963	0,8220922	0,8079225	0,8031810	0,8355329	0,8222543	0,8169371	0,8182398	0,7865966	0,7981062	0,7960115	0,7820114	0,7929721	0,7880992	
new-thyroid1	0,9464435	0,9459701	0,9488077	0,9459701	0,9488077	0,9488077	0,9460104	0,9488077	0,9318150	0,9459701	0,9488077	0,9459701	0,9488077	0,9488077	0,9634362	
new-thyroid1	0,9444699	0,9711371	0,9739747	0,9711371	0,9354764	0,9180506	0,9444699	0,9501049	0,9473075	0,9711371	0,9711371	0,9711371	0,9618958	0,9326791	**0,9767720**	
shuttle-c2-vs-c4	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	
yeast-1_vs_7	0,5501373	0,6237210	0,5102067	0,6259848	0,5258750	0,4976778	0,5967418	0,5903342	0,6330213	0,7318468	0,6452269	0,7717968	0,6376196	0,5806476	0,6359115	
yeast-2_vs_4	0,8907907	0,8782067	0,8751518	0,9075828	0,8617440	0,8637303	0,9091402	0,8675573	0,8833942	0,9361952	0,8809326	0,9152715	0,9092325	0,8653663	0,9079210	
yeast-2_vs_8	0,7294636	0,7945033	0,7952740	0,7924135	0,7342829	0,7314269	0,7258253	0,7316233	0,7267722	0,7848560	0,7641815	0,7848541	0,7240719	0,7603858	0,7273755	
MEDIA	**0,8014539**	**0,8507126**	**0,8101482**	**0,8472200**	**0,7796861**	**0,7818837**	**0,8219323**	**0,8071913**	**0,8185319**	**0,8526334**	**0,8184313**	**0,8418430**	**0,8231413**	**0,8012220**	**0,8108521**	

Continuación tabla de resultados

		ENSEMBLE + CHC + KNN														
		k = 5														
		IS			CS		IS-CS			MS			MS-CS			
DATASET	EBUS	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	EBUS	REDUCCI	PRECISIO	EBUS	
cleveland-0_vs_4	0,7668432	0,8932172	0,7089938	0,9201804	0,6717085	0,7216366	0,8088285	0,8115648	0,7638003	0,9002832	0,7656513	0,8851338	0,7565478	0,8627691	0,5707909	
ecoli-0-1-3-7_vs_2-6	0,6657005	0,7314163	0,6765776	0,7338403	0,5395948	0,6797246	0,7282028	0,6765776	0,6696242	0,7314163	0,6778811	0,7277462	0,7295188	0,7369997	0,7282028	
ecoli-0-1-4-6_vs_5	0,8504422	0,8906084	0,8769454	0,8993559	0,7966717	0,8275311	0,8474493	0,8804679	0,8542443	0,9150396	0,8752718	0,8696373	0,8939062	0,8396698	0,8768743	
ecoli-0-1-4-7_vs_2-3-5-6	0,8558171	0,8615830	0,8556537	0,8626480	0,8200597	0,8350862	0,8072169	0,8516220	0,8416948	0,8682445	0,8751287	0,8599083	0,8267615	0,8251129	0,8583220	
ecoli-0-1-4-7_vs_5-6	0,8423790	0,8904699	0,8478714	0,8921886	0,8331854	0,7983824	0,8122099	0,8320714	0,8230446	0,8827918	0,8302522	**0,8931544**	0,8432304	0,8139817	0,8882052	
ecoli-0-1_vs_2-3-5	0,8605088	0,8929029	0,8639192	0,8746140	0,7574907	0,8298550	0,8546190	0,7385542	0,7687356	0,8699084	0,8881097	0,8767071	0,7430867	0,8133864	0,8667601	
ecoli-0-1_vs_5	0,8222709	0,8948390	0,8792741	0,9289382	0,8862425	0,8263003	0,9044523	0,8528424	0,8242976	0,9148363	0,8367098	0,9504643	0,8222709	0,8508629	0,8206546	
ecoli-0-2-3-4_vs_5	0,8366272	0,8445251	0,8694167	0,8886858	0,8530450	0,8800495	0,8384938	0,8068445	0,8703183	0,9200249	0,8669927	0,8669927	0,8645337	0,9040801	0,8461870	
ecoli-0-2-6-7_vs_3-5	**0,8708539**	0,8482926	0,7800200	0,8576239	0,7640458	0,7418816	0,7937792	0,8005285	0,8561190	0,8430933	0,8380993	0,8528243	0,8553726	0,7718193	0,8440359	
ecoli-0-3-4-6_vs_5	0,8678434	0,8886171	0,8480109	0,9199856	0,8531995	0,8198971	0,8638380	0,8508099	0,8683204	0,9121182	0,8456542	0,8663603	0,8496938	0,8484532	0,9070319	
ecoli-0-3-4-7_vs_5-6	0,8508302	0,8954753	0,8570926	0,8757535	0,8414328	0,8395195	0,8952965	0,8629589	0,8797300	0,8955966	0,8799156	**0,9014628**	0,8931527	0,8760554	0,8713093	
ecoli-0-3-4_vs_5	0,7911994	**0,9171770**	0,8218790	0,8806013	0,8272749	0,8243729	0,8700864	0,8561566	0,8438663	0,8309163	0,8536628	0,9070215	0,8960915	0,8517560	0,7951903	
ecoli-0-4-6_vs_5	0,8514438	0,9193024	0,8671825	0,9201648	0,8196227	0,8698640	0,8911982	0,8511795	0,8358503	0,9137441	0,8435114	0,8643835	0,8651773	0,8776099	0,8650694	
ecoli-0-6-7_vs_3-5	0,8231822	0,8591226	0,8668417	0,8636389	0,8329095	0,8312522	0,8797006	0,8617099	0,8864896	0,8543382	0,8620866	0,8565743	0,8811262	0,8594329	0,8820100	
ecoli-0-6-7_vs_5	0,8144631	0,8814363	0,8242001	0,8619555	0,8065255	0,8043467	0,8235705	0,8209885	0,8183713	0,8835163	0,8215830	0,8619555	0,8184064	0,8234848	0,8169572	
ecoli-0_vs_1	0,9731766	0,9804517	0,9804517	0,9804517	0,9766551	0,9804517	0,9766551	0,9768478	0,9731766	0,9804517	0,9804517	0,9804517	0,9766551	0,9804517	**0,9839303**	
ecoli1	0,8453426	0,8503017	0,8403614	0,8571965	0,8167567	0,8403433	0,8561077	0,8410156	0,8562104	0,8934028	0,8625701	0,8612550	0,8506531	0,8394898	0,8614298	
ecoli2	0,9066815	**0,9319683**	0,9151881	0,9134075	0,8674759	0,8830320	0,9031235	0,8803569	0,8771907	0,9086404	0,9151881	0,9196612	0,9049560	0,9171930	0,8873112	
ecoli3	0,8222459	0,8381832	0,8171788	0,8340316	0,8141308	0,8172984	0,8166616	0,8483667	0,8164843	0,8686141	0,8477376	0,8402310	0,8368202	0,8155954	0,8277501	
ecoli4	0,8778028	0,9245922	0,8505995	0,9312167	0,8873102	0,8267675	0,8843473	0,9123200	0,9093333	0,9361279	0,9091781	**0,9391149**	0,9025818	0,8873322	0,9375084	
glass-0-1-2-3_vs_4-5-6	0,9231358	0,9362640	**0,9592134**	0,9456700	0,8799368	0,8858697	0,9269885	0,8985556	0,9476043	0,9430282	0,9480446	0,9556233	0,9331063	0,8920040	0,9090949	
glass-0-1-4-6_vs_2	0,5406966	0,7156909	0,5079069	0,5347614	0,4105928	0,5565822	0,6443191	0,6500439	0,5393540	0,6666294	0,6840986	0,6674130	0,6723823	0,5802267	0,5740828	
glass-0-1-5_vs_2	0,6413501	0,6829688	0,6537615	0,6819515	0,6280651	0,5242441	0,6487497	0,6617285	0,6048089	0,6754141	0,6783254	0,6629167	0,6335053	0,5146479	0,6261379	
glass-0-1-6_vs_2	0,4014167	0,5669528	0,5085438	0,5715762	0,2547051	0,5055877	0,5310057	0,5091802	0,5366364	0,6161223	0,5376860	0,6119133	0,5731904	0,5041488	0,5306762	
glass-0-1-6_vs_5	0,7942443	0,9736100	0,9736100	0,9736100	0,7942443	0,7942443	0,7942443	0,7942443	0,7942443	0,9736100	0,9736100	0,9736100	0,7942443	0,7942443	0,7942443	
glass-0-4_vs_5	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	0,9350705	**0,9936492**	0,9350705	**0,9936492**	0,9350705	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	**0,9936492**	
glass-0-6_vs_5	0,9258122	**0,9949359**	**0,9949359**	**0,9949359**	0,9363572	0,9363572	0,9363572	0,9363572	0,9363572	**0,9949359**	**0,9949359**	**0,9949359**	**0,9949359**	0,9363572	0,9363572	
glass0	0,8066909	0,8272924	0,8234221	0,8328427	0,7613624	0,7690104	0,8292143	0,7931890	0,7767776	0,8339568	0,8229230	0,8232657	0,8147119	0,8107299	0,8443692	
glass1	0,7831565	**0,8152902**	0,7580400	0,7864989	0,7317799	0,7574443	0,7821390	0,7373066	0,7751376	0,7979143	0,8125202	0,7628313	0,8010509	0,7642341	0,7787871	
glass2	0,4947383	0,4432926	0,4645229	0,3400060	0,5785228	0,4675828	0,6088152	0,3525804	0,4954398	0,5524854	0,4571671	0,4432194	0,6044872	0,3495807	0,5296925	
glass4	0,7006458	0,7343028	0,7470844	0,7420203	0,7124839	0,7537903	0,7436220	0,7145381	0,7394331	0,7394331	0,7445686	0,7394331	0,7441264	0,7170884	0,7031941	
glass5	0,7340282	0,9876679	0,7901219	0,9876679	0,7365132	0,7315433	0,9876679	0,9315433	0,9876679	0,9876679	0,7901219	0,9876679	0,9876679	0,7315433	**0,9926069**	
glass6	0,9203255	0,9236766	0,9177357	0,9237128	0,9228444	0,9177704	0,9238392	0,9228096	0,9319560	0,9158435	0,9151427	0,9266796	0,9240415	0,9177704	0,9203255	
haberman	0,6109924	0,6411136	0,5973503	0,6003213	0,6212906	0,6212906	0,6328554	0,6108972	0,6120983	0,6082874	0,5905685	0,6004464	0,6123661	0,6253184	0,6196334	
iris0	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	
led7digit-0-2-4-5-6-7-8-	0,7880992	0,8017454	0,8245383	0,8079225	0,8031810	0,8186255	0,8174992	0,8207183	0,8081567	0,8046399	0,8248979	0,8201987	0,8046174	0,8324875	0,8151094	
new-thyroid1	0,9634362	0,9738941	0,9488077	0,9459701	0,9488077	0,9634362	0,9488077	0,9488077	0,9634362	0,9621033	0,9488077	0,9488077	0,9488077	0,9488077	0,9634362	
new-thyroid1	**0,9767720**	0,9682995	0,9739747	0,9711371	0,9352689	0,9326791	0,9498974	0,9473075	0,9565086	0,9682995	0,9739747	0,9711371	0,9501049	0,9473075	0,9562608	
shuttle-c2-vs-c4	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	
yeast-1_vs_7	0,6359115	0,6464340	0,4868706	0,6962941	0,5241574	0,5493752	0,5794904	0,5402507	0,6410218	0,7590881	0,7341311	0,7317943	0,6403122	0,5596190	0,6562784	
yeast-2_vs_4	0,9079210	0,9157567	0,8976308	**0,9413842**	0,8496012	0,8726988	0,9291072	0,8328775	0,9117874	0,9324253	0,9039164	0,9358025	0,9246135	0,8643232	0,9223994	
yeast-2_vs_8	0,7273755	0,7607383	0,7941504	0,7908635	0,7335036	0,7337476	0,7238067	0,7335204	0,7308524	0,8063044	0,7594253	0,7808295	0,7540425	0,7641642	0,7232969	
MEDIA	0,8108521	0,8554237	0,8203158	0,8511669	0,7846039	0,7988776	0,8314064	0,8127054	0,8202616	0,8629686	0,8369973	0,8548322	0,8358629	0,8103220	0,8265452	

Continuación tabla de resultados

	ENSEMBLE + KNN			E + CHC		E
DATASET	k = 1	k = 3	k = 5	REDUCCIO	PRECISION	
cleveland-0_vs_4	0,6700340	0,6690292	0,7266553	0,8893484	0,8877305	0,8805087
ecoli-0-1-3-7_vs_2-6	0,4765776	0,6815512	0,6815512	0,6827986	0,8425845	0,6853742
ecoli-0-1-4-6_vs_5	0,8241341	0,8183667	0,8484769	**0,9485447**	0,9070450	0,8851141
ecoli-0-1-4-7_vs_2-3-5-6	0,8355921	0,8423412	0,8438196	**0,8768175**	0,8675644	0,8537333
ecoli-0-1-4-7_vs_5-6	0,8317497	0,7976343	0,8361550	0,8800436	0,8764662	0,8719821
ecoli-0-1_vs_2-3-5	0,7620759	0,8058889	0,8789205	0,8439926	**0,9104035**	0,8743938
ecoli-0-1_vs_5	0,7948798	0,8090551	0,8052894	**0,9509733**	0,9150766	0,9019252
ecoli-0-2-3-4_vs_5	0,8486983	0,8380654	0,8694167	0,8839121	0,9054023	0,8716766
ecoli-0-2-6-7_vs_3-5	0,7117209	0,7582489	0,8425485	0,8496029	0,8472950	0,8381584
ecoli-0-3-4-6_vs_5	0,9070319	0,8771954	0,8480438	0,8647078	0,8911604	0,8719985
ecoli-0-3-4-7_vs_5-6	0,8394777	0,8629589	0,8629589	0,8839431	0,8467583	0,8818482
ecoli-0-3-4_vs_5	0,8272749	0,8218790	0,8536628	0,8996574	0,8835969	0,8866148
ecoli-0-4-6_vs_5	0,9044378	0,8454894	0,8486273	0,8818197	0,8129073	0,8240746
ecoli-0-6-7_vs_3-5	0,8200005	0,8379736	0,8693576	0,8531076	0,8730053	0,8449927
ecoli-0-6-7_vs_5	0,7775518	0,8052513	0,8319125	0,8553822	0,7787932	0,8743165
ecoli-0_vs_1	0,9766551	0,9731766	0,9804517	0,0092848	0,0092848	0,0092848
ecoli1	0,8280037	0,8368084	0,8570464	0,8879296	**0,9032307**	0,8991007
ecoli2	0,8953816	0,9184085	0,9078173	0,8898345	0,8933111	0,9164546
ecoli3	0,7729675	0,7465667	0,8279581	**0,8736029**	0,8668303	0,8731378
ecoli4	0,8253874	0,8251738	0,8571819	0,9282991	0,8977432	0,8961785
glass-0-1-2-3_vs_4-5-6	0,9205866	0,9261107	0,9385856	0,9139988	0,9191308	0,9405641
glass-0-1-4-6_vs_2	0,4128760	0,5578066	0,6551395	0,7134702	0,6747932	**0,7487649**
glass-0-1-5_vs_2	0,6092243	0,6137857	0,6447213	0,6640010	0,6921415	0,6479471
glass-0-1-6_vs_2	0,5053131	0,5089543	0,5034070	**0,7022184**	0,5881398	0,5976985
glass-0-1-6_vs_5	0,7913238	0,6738299	0,8680315	0,6959566	0,9410260	0,9410260
glass-0-4_vs_5	0,9350705	**0,9936492**	0,9936492	0,9445105	**0,9936492**	**0,9936492**
glass-0-6_vs_5	0,9363572	**0,9949359**	**0,9949359**	0,9207733	0,9100534	0,9208723
glass0	0,7757818	0,8062452	0,8197722	0,8128230	0,8085664	0,8101268
glass1	0,7701928	0,7521852	0,7834875	0,7686977	0,7802558	0,7632799
glass2	0,3475226	0,4587188	0,4571671	0,6640015	0,7508891	**0,7748667**
glass4	0,6812701	0,7200362	0,7470844	**0,9033652**	0,8625263	0,8893651
glass5	0,7365132	0,7365132	0,7901219	0,9353855	0,9472773	0,9472773
glass6	0,9228096	0,9202908	0,9202908	0,8918422	0,9054977	0,9078541
haberman	0,5303758	0,5487329	0,6060895	0,6650553	0,6525228	**0,6650965**
iris0	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**	**0,9897367**
led7digit-0-2-4-5-6-7-8-	**0,8385758**	0,8336983	0,8325696	0,7912183	0,7925549	0,8009655
new-thyroid1	0,9488077	0,9309626	0,9488077	0,9530388	0,9469225	0,9553968
new-thyroid1	0,9473075	0,9447177	0,9593462	0,9565086	0,9685472	0,9597743
shuttle-c2-vs-c4	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**	**1,0000000**
yeast-1_vs_7	0,5307532	0,6014207	0,6025826	0,7596065	0,7440504	**0,7882654**
yeast-2_vs_4	0,8326340	0,8551957	0,8634692	0,9122063	0,9048303	0,9281742
yeast-2_vs_8	0,7647831	0,7672686	0,7682074	0,7852361	0,8092607	0,7512681
MEDIA	**0,7823202**	**0,7977585**	**0,8229775**	**0,8327917**	**0,8380610**	**0,8372104**

- **ENSEMBLE + CHC + KNN:** Hace referencia a la familia de resultados obtenidos tras aplicar el algoritmo CHC para la reducción de instancias y/o características. En este caso, la fase de agregación es realizada con el algoritmo kNN.
- **ENSEMBLE + KNN:** Hace referencia a la familia de resultados obtenidos tras agregar los resultados obtenidos en el ensemble con el algoritmo kNN.

Las restantes 4 columnas indican la fase de comparación de resultados en la que se encuentra el proceso (Aunque en la primera se añada información de la configuración de cada método de las familias). Tal y como se puede ver en la figura, estas fases son 4: “Tipo de fitness”, “Valor de k”, “IS | MS – IS-CS | MS-CS” y “Comparación Métodos”. Más adelante explicaremos en qué consiste cada una de estas cuatro fases.

Los valores contenidos en los recorridos (figura 15) indican los rangos obtenidos en el test estadístico correspondiente, Wilcoxon para el caso de comparación entre dos métodos y Friedman en caso de ser más, y pertenecen al método del que provienen.

Los valores contenidos fuera de los recorridos (figura 14) hacen referencia a los p_{valor} distintos de cada método a la misma altura que él valor, obtenidos con los métodos estadísticos. En los casos en los que el p_{valor} esté en negrita significa que es menor que 0.05 por lo que el método ganador es estadísticamente mejor al método al que haga referencia dicho p_{valor}.

En la comparación final se marca con el “tag” “GANADOR” al método que obtiene menor número de rangos en el test de Friedman para las comparaciones finales entre, los mejores métodos entre los propuestos, y el ensemble original.

Por último los colores en sí no indican nada. No significa que todos los métodos con una serie de configuraciones estén dibujados con el mismo color. Sirven para orientar al lector y hacer más fácil y visual la lectura de los resultados obtenidos. El punto más importante es que en los casos en los que se juntan varios colores/caminos, la comparación entre los métodos devuelve como mejor resultado el color/camino que sigue en las siguientes fases.

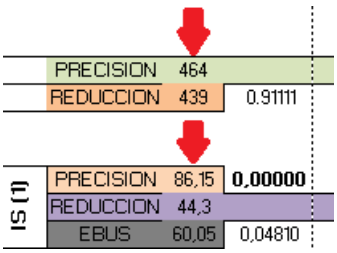


Figura 15 Ejemplo de rangos

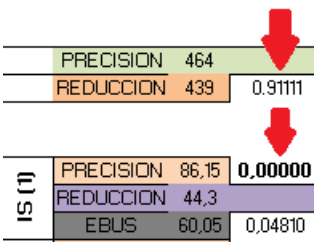


Figura 14 Ejemplo p_{valor}

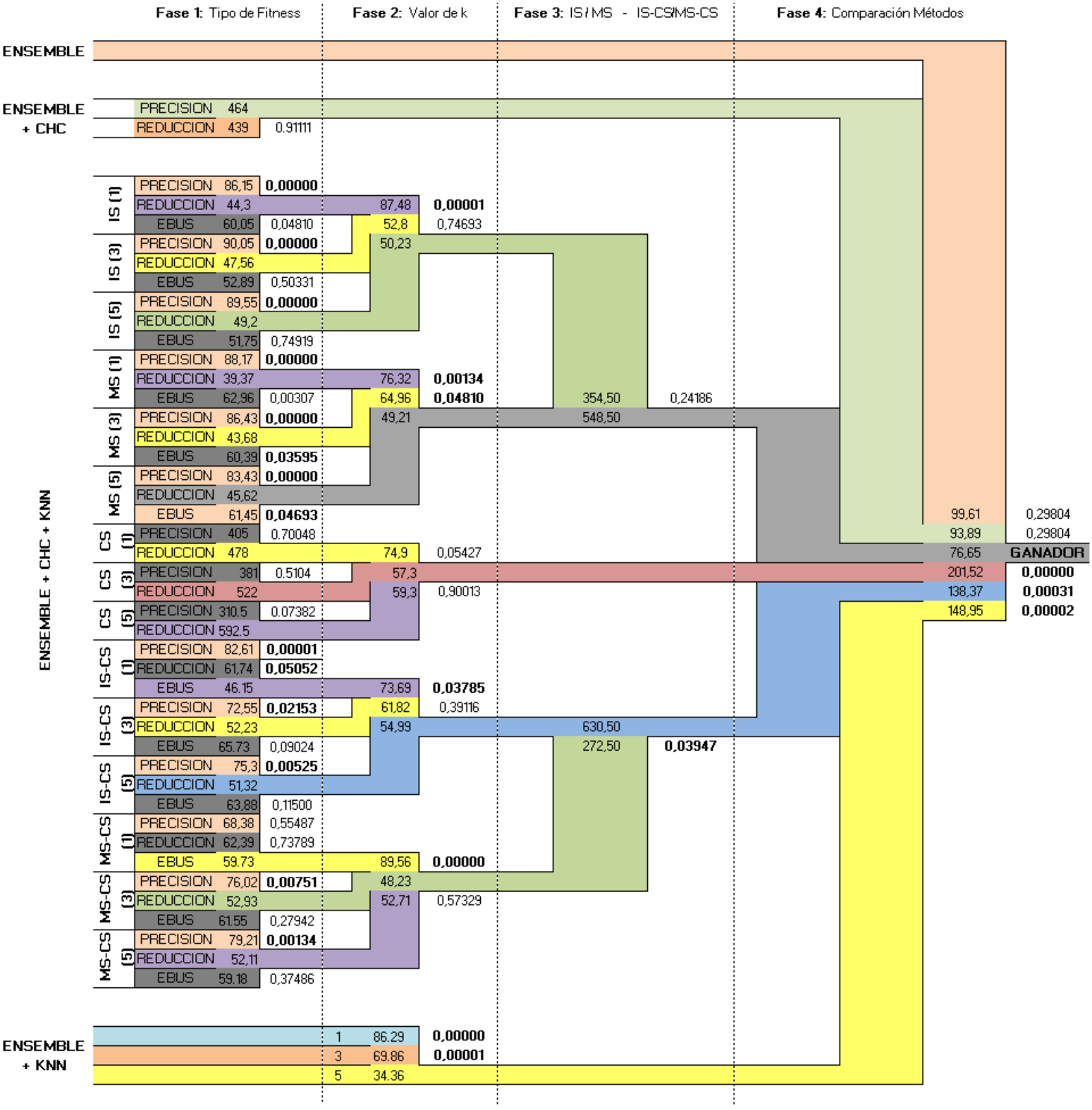


Figura 16 Resumen resultados UNDERBAGGING con 40 clasificadores

5.3 Metodología en la comparación de métodos

Debido a la gran cantidad de métodos propuestos, a la hora de comparar los resultados obtenidos es necesario separar por fases las comparaciones. Esto es debido a que el resultado obtenido de aplicar el test estadístico de Friedman comparando todos los métodos de golpe no sería para nada fiable. Como ejemplo de esto, más adelante se realiza una comparación con el test de Wilcoxon entre el método ganador y el original y se muestra como el método ganador sí es estadísticamente mejor al método original, al contrario de lo que muestra el test de Friedman de la fase de la comparación de métodos.

Las fases en las que descomponemos la comparación de métodos son las siguientes:

1. **Tipo de fitness:** En esta primera fase la comparación en las familias de “Ensemble + CHC + KNN” y “Ensemble + CHC” se basa en comparar las diferentes formas de calcular el fitness en el algoritmo genético. En los casos de la familia “Ensemble + CHC” y “CS” de la familia “Ensemble + CHC + KNN” solamente utilizamos los métodos Reduccion y Precision. Esto es debido a que el método del Ebus favorece el balanceo en la cantidad de las diferentes clases. Este método no tiene sentido aplicarlo en métodos que únicamente reducen clasificadores. En el resto de métodos de la familia “Ensemble + CHC + KNN” se utilizan en las comparaciones los tres tipos de fitness. Para la familia “Ensemble + KNN” y para el caso del ensemble original no tiene sentido aplicar esta primera fase de comparaciones.
2. **Valor de k:** En la segunda fase la comparación tiene como objetivo determinar para los métodos ganadores de la fase anterior (incluyendo en este caso la familia “Ensemble + KNN”, aunque no eran comparados antes) el valor de k que mejores resultados obtiene. Así, las comparaciones serán entre los métodos con iguales configuraciones exceptuando el valor de k.
3. **IS | MS – IS-CS | MS-CS:** En esta fase, la comparación tiene como objetivo determinar cuáles de los métodos basados en la reducción de instancias, ya sea solo de instancias o también de características, son mejores. Es decir, vamos a comparar los métodos de reducción de instancias, *Instance Selection y Majority Selection*, entre sí y los métodos de reducción de instancias y características o clasificadores en este caso entre sí, *Instance Selection-Clasifier Selection y Majority Selection-Clasifier Selection*.
4. **Comparación métodos:** En la última fase la comparación se realizará entre los mejores métodos obtenidos hasta el momento.

El método con menor número de rangos será el señalado como “GANADOR”, ya que utilizamos el test estadístico de Friedman debido a que tratamos con más de dos métodos.

5.4 Resultados obtenidos

Como los resultados obtenidos son muy numerosos, vamos a explicarlos uno a uno de forma que se entienda el porqué de cada uno de ellos y lo que significan. Para ello vamos a explicarlos siguiendo las cuatro fases explicadas en el apartado anterior.

Fase 1: Tipo de fitness

Empezando por el ensemble original, debido a que no se aplica ningún tipo de fitness, pasa directamente a la siguiente fase (figura 17).

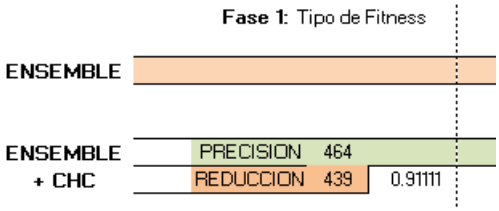


Figura 17 Fase 1: Ensemble y "Ensemble + CHC"

Continuando con la familia “Ensemble + CHC”, tal y como se puede ver en la figura 16, únicamente se contemplan dos posibilidades: Reduccion y Prediccion. Debido a que solo son dos los métodos comparados, el test estadístico aplicado es el test de Wilcoxon. Como resultado podemos observar que ninguno de los dos métodos es estadísticamente mejor que el otro. Si es cierto que el método de la precisión obtiene mayor número de rangos que el método de la reducción pero el p_{valor} no es menor que 0.05. Por tanto el método que accede a la siguiente fase es el método con la Precision pero eso no significa que sea estadísticamente mejor, solo que para los datasets probados ha obtenido mejores resultados. Esto puede ser debido a que en la reducción al darle cierto valor a la reducción de clasificadores, trata de reducir demasiados y a la hora de comprobar su efectividad en el conjunto de test los resultados empeoren.

Siguiendo con la familia “Ensemble + CHC + KNN”, podemos ver en la figura 18 que las diversas comparaciones son realizadas entre los diferentes métodos para calcular el fitness. Estas comparaciones están separadas por el método utilizado en la reducción y por el valor de k por lo que las trataremos una por una:

- **IS (1):** En este caso se aplica el estadístico de Friedman. Podemos ver como el método de la reducción es el que mejores resultados ha obtenido. Esto es debido a que al eliminar instancias (aunque elimine de ambas clases elimina más de la mayoritaria por el hecho de que ya de por sí son muchas más) de la clase mayoritaria mejora los resultados obtenidos en la clasificación. Además, como los p_{valor}-es son menores que 0.05 el método de la reducción es estadísticamente mejor que los otros dos para este caso.
- **IS (3):** En este caso podemos ver como el método de la reducción vuelve a ser el ganador, aunque para este caso solo es estadísticamente mejor al método de la precisión. Esto es debido a que como en el método del Ebus se intenta balancear el número de instancias de cada clase, teniendo en cuenta una vecindad mayor tiene más posibilidades de acertar que en el caso de la vecindad reducida a un único elemento.
- **IS (5):** Este caso es exactamente igual al caso anterior, solo que teniendo en cuenta una vecindad aún mayor por lo que la diferencia de rangos entre la reducción y el ebus es menor.
- **MS (1):** En este caso, vuelve a ganar la reducción siendo estadísticamente mejor que los otros dos métodos. Esto es debido a que la precisión no aumenta el fitness por reducir instancias, lo cual es necesario, y a que el ebus trata de balancear el número de instancias de cada clase en vez de reducir al máximo el número de instancias de la clase mayoritaria tal y como hace la reducción.
- **MS (3):** Este caso vuelve a ser igual al caso anterior, por lo que gana la reducción.
- **MS (5):** Este caso vuelve a ser igual a los dos casos anteriores, por lo que vuelve a ganar la reducción.
- **CS (1):** En este caso, no se puede decir que el método estadísticamente mejor sea la reducción, pero para este conjunto de datasets sí que obtiene mejores resultados. Esto puede ser por que al darle cierta prioridad a la reducción de clasificadores, elimine algunos clasificadores que no clasifiquen

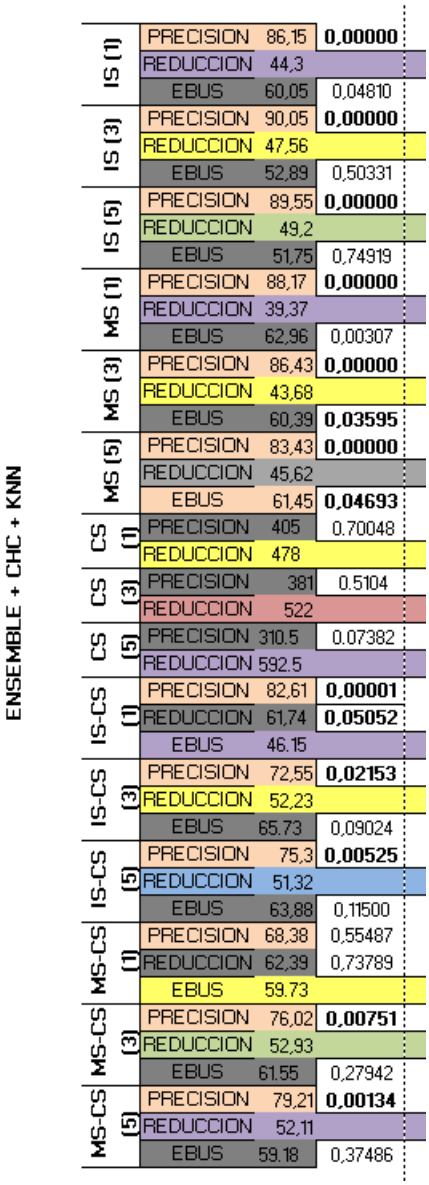


Figura 18 Fase 1: "Ensemble + CHC + KNN"

correctamente la instancia minoritaria, lo que hace que la tasa de acierto de dicha clase aumente y esto aumente la Gmean obtenida como resultado.

- **CS (3):** Este caso es igual al caso anterior, por lo que gana la reducción pero no podemos afirmar que sea estadísticamente mejor que la precisión.
- **CS (5):** Este caso es similar a los dos anteriores. Gana la reducción y no podemos afirmar que sea estadísticamente mejor a la precisión. Por otra parte, podemos ver como la diferencia entre los rangos va en aumento y el p_{valor} va reduciendo. Esto indica que el método de la reducción mejora más cuantos más vecinos tengamos en cuenta en el algoritmo kNN.
- **IS-CS (1):** En este caso gana el ebus siendo estadísticamente mejor que la precisión y con un p_{valor} próximo a 0.05 para el caso de la reducción. Esto es debido a que teniendo en cuenta una vecindad reducida, es preferible tratar de balancear el número de instancias de ambas clases a eliminar instancias sin tener en cuenta la cantidad de cada una de las clases, tal y como pasa con la reducción.
- **IS-CS (3):** En este caso gana la reducción siendo estadísticamente mejor que la precisión y no siendo estadísticamente mejor que el ebus. Esto es debido a que, como hemos visto en el caso anterior, a la hora de ejecutar la selección de instancias es útil el tratar de balancear la cantidad de instancias de cada clase, aunque en este caso como la vecindad aumenta ya no sea tan necesario.
- **IS-CS (5):** Este caso es igual al caso anterior por lo que gana la reducción siendo estadísticamente mejor que la precisión y sin ser estadísticamente mejor que el ebus.
- **MS-CS (1):** En este caso el método con menor número de rangos es el ebus por lo que es el ganador aunque no mejora estadísticamente a los otros dos métodos. Aún así, si observamos los rangos de los otros dos métodos podemos ver como la reducción es relativamente próxima por lo que concluimos que no existe mucha diferencia entre esta y el ebus para una vecindad reducida. Esto puede deberse a que, al igual que con IS-CS(1), el balancear el número de instancias de cada clase da mejor resultado que reducir todo tipo de instancias para vecindades reducidas.
- **MS-CS (3):** En este caso gana la reducción siendo estadísticamente mejor que la precisión y sin serlo del ebus. Esto es debido a que es preferible reducir el mayor número de instancias posibles a tratar de balancear la cantidad de las mismas sobre cada clase, siempre y cuando se trabaje con una vecindad mediana o amplia.

- **MS-CS (5):** Este caso es similar al anterior por lo que gana la reducción siendo estadísticamente mejor que la precisión y sin serlo del ebus.

Como podemos ver, en esta familia de métodos nunca es mejor utilizar la precisión como fitness a utilizar cualquiera de los fitness que tratan de reducir la cantidad de instancias, ya sea balanceandolas con ebus o reduciendo sin tener en cuenta las clases con la reducción.

Por último, la familia de “Ensemble + KNN”. En este caso, a esta familia no hay que aplicar de momento comparación alguna ya que no se utiliza ningún tipo de fitness durante el proceso por lo que pasan a la siguiente fase todos los métodos que pertenecen a esta familia.

Fase 2: Valor de k

En esta fase, al método del ensemble y al método ganador de la fase 1 de la familia de métodos “Ensemble + CHC” no hay que aplicar comparativa alguna ya que no utilizan ningún valor de k. Por tanto, pasan a la siguiente fase.

Siguiendo con la familia “Ensemble + CHC + KNN” podemos ver como las comparaciones son entre los ganadores de la fase 1 de esta misma familia, pero entre los métodos que comparten el tipo de reducción aplicado. Esto es para seleccionar el mejor valor de k para esos métodos.

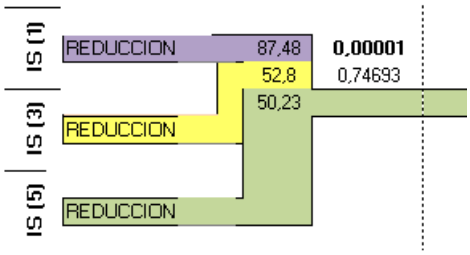


Figura 19 Fase 2: IS

Tal y como se puede ver en la figura 19, los resultados de la comparación entre los ganadores de la fase 1 de Instance Selection con sus diferentes valores de k son que el mejor valor es 5. Gracias al p_{valor} podemos ver que es estadísticamente mejor al valor de k 1 y que no hay diferencias significativas con el valor de k 3. Esto es debido a que el tratar con una vecindad demasiado reducida estaba perjudicando los resultados obtenidos debido al ruido, mientras que al ampliar la vecindad han sido mejorados.

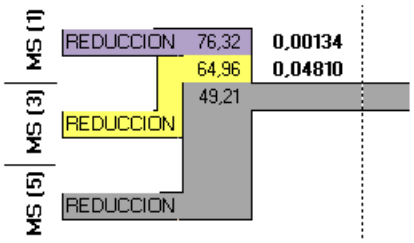


Figura 20 Fase 2: MS

En la figura 20 se muestran los resultados de la comparación entre los ganadores de la fase 1 de Majority Selection con sus diferentes valores de k. En este caso se vuelve a observar como la mayor vecindad es el mejor de los tres metodos, aunque en este caso es mejor estadísticamente a ambos. Esto se debe a que al reducir instancias de la clase mayoritaria, es mejor contar con una mayor vecindad para clasificar que contar con una vecindad reducida que puede verse afectada más facilmente por las instacias de la clase opuesta, la cual no reducimos.

En la figura 22 se muestran los resultados obtenidos en la fase 2 para Clasifier Selection. Como podemos ver, en este caso gana un valor de vecindad mediano, el 3. Aún así, no podemos decir que es estadísticamente mejor que el método con valor k 5. Para el caso de k 1 podemos observar como el p_{valor} es muy próximo a 0.05 lo que demuestra que sí es peor que los otros dos. Esto es debido a que con una vecindad muy pequeña, el reducir clasificadores no da muy buen resultado.

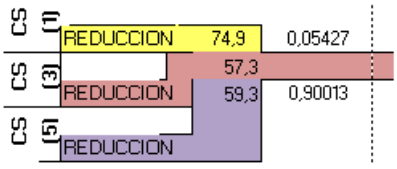


Figura 22 Fase 2: CS

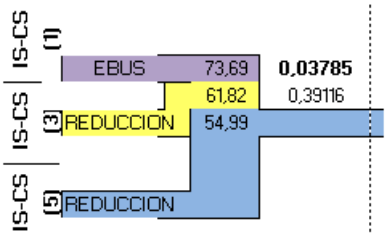


Figura 21 Fase 2: IS-CS

Como podemos observar en la figura 21, tenemos los resultados de la segunda fase para IS-CS. En este caso gana el método con valor de k 5 siendo mejor estadísticamente que el k 1 y sin ser estadísticamente mejor pero sí con mejores resultados que k 3. Esto es debido a que al reducir tanto instancias como cromosomas es mejor tener en cuenta una mayor vecindad ya que evita malas clasificaciones debido a la reducida cantidad de ellos.

En la figura 24 podemos observar los resultados obtenidos con MS-CS. En este caso gana el valor de k mediano, siendo mejor estadísticamente que el k pequeño y sin serlo del k grande. Esto quiere decir que tanto el k mediano como el grande funcionan bien para los casos en los que eliminamos instancias de la clase mayoritaria con MS.

Esto es debido a que como solo eliminamos instancias de la clase mayoritaria, los resultados en la clase minoritaria mejoran, lo que permite que el valor de la vecindad mediano gane al mayor.

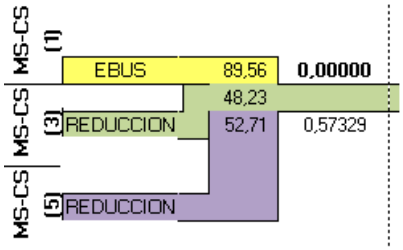


Figura 24 Fase 2: MS-CS

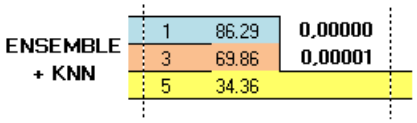


Figura 23 Fase 2: "Ensemble + KNN"

En la figura 23 podemos ver los resultados obtenidos para la familia de métodos “Ensemble + kNN”. En este caso, como no se aplica reducción alguna, cuanto mayor sea la vecindad mejores son los resultados. Por ello, el valor de k ganador es el 5 siendo mejor estadísticamente que los otros dos propuestos.

Fase 3: IS | MS – IS-CS | MS-CS

En esta fase tratamos de comparar los métodos de reducción de instancias con y sin balanceo para ver cuáles son mejores para comparar en la fase final. En la figura 25 se pueden ver los resultados obtenidos.

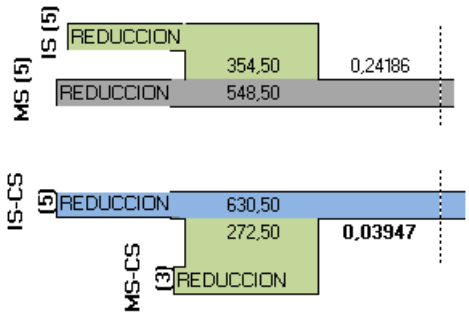


Figura 25 Fase 3

Como podemos ver, para el caso de IS | MS tenemos que el mejor de ambos es MS pero no estadísticamente mejor. Esto se debe a que en este caso solamente reduce instancias de la clase mayoritaria lo que no perjudica demasiado los resultados obtenidos en la minoritaria.

Para el caso IS-CS | MS-CS en cambio, obtenemos el resultado contrario. Como llevamos a cabo una reducción de clasificadores, de los cuales seguramente exista alguno que no clasifique bien la clase minoritaria, es preferible reducir también instancias de la clase minoritaria que solo reducir de la mayoritaria. Hay que destacar que el método IS-CS mejora estadísticamente al otro método.

Fase 4: Comparación métodos

Tal y como podemos observar en la figura 26, el método ganador es de la familia “Ensemble + CHC + KNN”, más concretamente el MS con k 5. Este método mejora estadísticamente a los métodos propuestos exceptuando al método de la familia “Ensemble + CHC” con precisión.

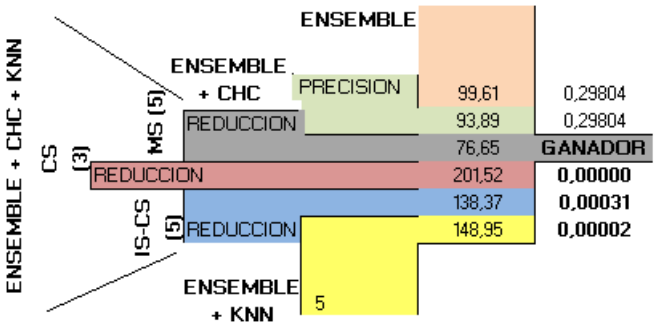


Figura 26 Fase 4

Tal y como hemos dicho, en los casos en los que tengamos más de dos métodos para comparar utilizamos el test estadístico Friedman. El problema de este test reside en que si el número de métodos a comparar es demasiado elevado sus resultados no son del todo correctos. Sí que son correctos en el sentido de que por orden de rangos se establece el ganador de entre todos los métodos y el orden de mejor a peor. En lo que no es preciso es en destacar si un método mejora estadísticamente a otro o no. Vamos a demostrar que el ganador entre todos los métodos propuestos es estadísticamente mejor que el método del ensemble original. Para ello, hemos hecho un test de Wilcoxon, el cual es más preciso que el de Friedman para comparar dos métodos distintos. Podemos ver el resultado obtenido en la siguiente figura.

	E+CHC+KNN(5_MS_REDUCCION)	ENSEMBLE	P
UNDERBAGGING-40	648	255	0.019056

Figura 27 Comparación Wilcoxon entre método ganador y ensemble original

Tal y como podemos ver en la figura, el método ganador es mejor que el ensemble original. Con unos valores de rangos de 648 para el ganador y 255 para el ensemble original, siendo esta una diferencia importante. De ahí surge el p_{valor} obtenido, 0.019056. Este valor es menor que 0.05 (al contrario que en el test de Friedman), por lo que el método ganador es estadísticamente mejor que el método original.

Debido a la longitud del estudio realizado, para evitar repetir información y hacer de forma más legibles las tablas de resultados obtenidas en los diferentes métodos con las diferentes cantidades de clasificadores, adjunto junto con esta memoria una carpeta llamada Anexo 1 compuesta de ficheros “.xlsx”, los cuales contienen las diferentes tablas de resultados para cada cantidad de clasificadores y cada método del ensemble para poder ser revisadas con el programa “Excel”.

6.- Conclusiones y Líneas Futuras

En este proyecto hemos propuesto una nueva forma de abordar el problema de la agregación de resultados obtenidos en los ensembles para los problemas de los conjuntos de datos no balanceados. El método original se basa en utilizar el voto. Nosotros, en cambio, proponemos una gran cantidad de métodos nuevos pero pueden ser resumidos en tres familias distintas:

- **Ensemble + CHC:** Esta familia de métodos, aunque en sí no modifica la forma de agregar las salidas obtenidas por el ensemble, sí modifica la cantidad de ensembles que componen el ensemble por lo que modifica los valores obtenidos en las salidas finales.
- **Ensemble + kNN:** Esta familia de métodos recurre al algoritmo kNN para agregar los resultados obtenidos por el ensemble.
- **Ensemble + CHC + kNN:** Esta familia de métodos recurre al algoritmo genético CHC para reducir el número de clasificadores que componen el ensemble y para reducir el número de instancias que componen el conjunto de datos de train. Una vez hecho eso, agrega los resultados obtenidos con el algoritmo kNN.

Para comparar los resultados obtenidos hemos llevado a cabo una serie de test estadísticos y los hemos analizado en profundidad. En resumen las conclusiones obtenidas son las siguientes:

- Algunos de los métodos propuestos obtienen mejores resultados en los datasets con los que hemos probado.
- Los nuevos métodos, ya que están basados en su mayor parte en métodos que reducen las instancias del conjunto de train y los clasificadores que componen el ensemble, además de obtener mejores resultados en lo que a clasificación respecta, obtienen un mejor rendimiento, un menor espacio ocupado y por tanto una mayor eficiencia.
- Al menos uno de los métodos propuestos (“Ensemble + CHC + kNN” con k = 5, Majority Selection y reducción) es estadísticamente mejor que el método tradicional del voto. Esto es, hemos hallado una forma de realizar la fase final de los ensembles, la de agregación de los resultados obtenidos, mejor que el voto. Como ejemplo, para el caso del algoritmo

UnderBagging con 40 clasificadores la media de los resultados obtenidos en cuanto a la media geométrica es **0.8372104**. Mientras que para nuestro método propuesto, la media en los diferentes datasets es **0.8629686**. Aquí queda reflejado que en los datasets con los que hemos tratado el nuevo método es mejor. Además, para comprobar si realmente es mejor o no, hemos empleado el test estadístico de Wilcoxon, a través del cual hemos confirmado que el nuevo método planteado mejora estadísticamente al método original. Cabe destacar que este método además de mejorar el resultado obtenido, reduce en gran medida el número de instancias de la clase mayoritaria del conjunto de train.

En lo que a líneas futuras se refiere, este proyecto tiene varias ya que los resultados obtenidos son más que favorables:

- La primera y más clara de todas ellas es la de realizar un estudio para comprobar si el aplicar nuevas técnicas en la fase de agregación de los ensembles, basadas en técnicas genéticas y en el algoritmo kNN, también mejoran los resultados obtenidos en conjuntos de datos balanceados.
- La segunda y también bastante clara, es la de realizar un estudio para comprobar si el aplicar nuevas técnicas en la fase de agregación de los ensembles, basadas en técnicas genéticas y en el algoritmo kNN, también mejoran los resultados obtenidos en conjuntos de datos multiclase.
- Otra de las líneas futuras es la de ampliar el estudio realizado a través de diversos puntos:
    - Cambiar la distancia Euclidea utilizada en el algoritmo kNN por otras distancias distintas.
    - Probar distintos valores de k.
    - Probar con un mayor o menor número de clasificadores en el ensemble.

7.- Bibliografía

[1] M. Galar, A. Fernández, E. Barrenechea, H. Bustince and F. Herrera, "A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches", *IEEE Transactions on systems, man, and cybernetics – part c: applications and reviews,* vol. 42, no. 4, 4 July 2012.

[2] R. Polikar, "Ensemble Based Systems in Decision Making", Circuits and Systems Magazine, IEEE (Volume:6 , Issue: 3), third quarter 2006.

[3] N. Ramakrishnan, "C4.5", Taylor & Francis Group, LLC, 2009.

[4] M. Steinbach and P. Tan, "kNN: k-Nearest Neighbours", Taylor & Francis Group, LLC, 2009.

[5] L. Eshelman, "The chc adaptative search algorithm: How to have safe serach when engaging in nontraditional genetic recombination", Foundations of Genetic Algorithms, pp. 265-283, 1991.

[6] J. Derrac, S. García and F. Herrera, "IFS-CoCo: Instance and Feature selection based on cooperative coevolution with nearest neighbor rule", ScienceDirect, Pattern Recognition, www.elsevier.com/locate/pr, 2010

[7] S. García and F. Herrera, "Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy", Massachusetts Institute of Technology, Evolutionary Computation, 2009

[8] S. A. Dudani, "The Distance-Weighted k-Nearest-Neighbor Rule", Hughes Research Laboratories, Malibu, 1975